IEEE MICRO

APRIL 1990

Chips, Systems, Software, and Applications

INNOVATIVE TECHNOLOGY IN THE FAR EAST

The TRON House Showcase

See About the Cover on page 6.

Special Feature: Memory Management Units

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Image from Ken Sakamura
Cover design: Design & Direction

IEEE MICRO

Volume 10 Number 2 (ISSN 0272-1732) April 1990

Published by the IEEE Computer Society

Departments

Call for papers, p. 25; Reader Interest/Service/Subscription cards, p. 64A; Change-of-Address form, p. 94

Features

Special Feature

The Intel Influence

PERFORMANCE.

We've built an extremely successful company on it. Involved in components, modules and systems, we depend on high achieving, performance-driven individuals to help us lead the microelectronics industry into the year 2000.

PLATFORM ARCHITECTURE GROUP

Join the team that will set the architecture directions for the premiere microprocessors and peripherals of the industry and take Intel's megatransistor technologies and apply them to the remainder of the computer architecture.

Extensive expertise is required in one of the following areas for these Technical Staff Members.

- Multiprocessing
- Simulations
- Cache and Memory Bus
- Virtual Machine Architecture
- Numerical Analysis
- Software Architecture
- I/O
- Performance Modeling

WORLD CLASS MICROPROCESSOR DESIGN

i860™ Microprocessor Architect/Manager-Next Generation

Lead and manage the architecture and microarchitecture of the next generation high performance microprocessor. You will be responsible for performance tradeoffs and implementing new features. Involves heavy group interaction with the implementation team and the software group. This position will require approximately 10 years on the architecture/implementation of large computer systems. Knowledge of compilers and OS's required. MSEE/CS or equivalent essential.

Architects

Requires at least 3-5 years dealing with system level architecture issues including 2 or more of the following: I/O, second level cache, multiprocessing, memory subsystems, compiler design, or pipelining. Familiarity with VLSI design and customer interface experience is a plus. MSEE or CS required.

Workstation and Graphic Systems Architects

Utilize your expertise in systems architecture and design for workstations or graphic systems while working closely with customers on the definition of their systems with the i860 and next generation microprocessor families. You will develop competitive analysis of other approaches to design these systems. You will also develop and deliver customer presentations on the use of the i860 microprocessor workstation, minicomputer or graphics system. Requires BS/MS EE or CS with 6+ years related experience.

CAD ENGINEERS

(Jr. and Sr. Positions)

Utilize your CAD or CMOS VLSI design experience for one of the following positions:

CAD Tool Evaluator

You will define and implement competitive analysis of CAD systems, tool and design methodologies, as well as evaluate commercial CAD tools in cooperation with designers. You will design and implement interfaces as necessary.

High Level Modeling Expert

You will participate in definition, evaluation/design, and implementation of a High Level Modeling System for Intel's design environment while interacting with designers, CAD engineers and vendors. Simulation, synthesis, or modeling of microprocessors is highly desired.

Both positions require BS/MSEE (Ph.D. desired) and 5+ years VLSI CMOS design and/or CAD expertise.

NEXT GENERATION HIGH PERFORMANCE PRODUCTS

Product Marketing Manager

Responsible for pricing, positioning and introducing all products, training the sales force, and performing competitive research and analysis. Requires an MBA with a technical background preferred and at least 4-5 years experience. Should have market development skills, industry understanding of microprocessor architectures, thorough understanding of the communication functions, such as dealing with the press, developing campaigns and merchandising activities.

Product Planning Manager

Will develop future product specifications using Intel product planning systems, manage the planning process, maintain product line strategy documents and facilitate product development teams. Requires an MSEE with experience in strategic planning and/or product definition for RISC products, caches, multichip modules or next generation high performance microprocessors. Computer system or microprocessor architectures preferred. Experience with customers soliciting requirements and feedback for product definition, market research, competitive analysis, and market segmentation is ideal.

For immediate consideration, please contact Becky Canary at (408) 765-3828, or send your resume to: **Intel Corporation, Dept. S100, P.O. Box 58121, Santa Clara, CA 95052-8121.** Intel Corporation is an equal opportunity employer and fully supports affirmative action practices. Intel also supports a drug-free workplace and requires that all offers of employment be contingent on satisfactory pre-employment drug test results.

intel®

IEEE Computer Society
PO Box 3014
Los Alamitos, CA 90720-1264
(714) 821-8380

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: $19 in addition to IEEE Computer Society or any other IEEE society member dues; $35 for members of other technical organizations. Back issues: $10 for members; $20 for nonmembers. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

* Submit six copies of all articles and special-issue proposals to Joe Hootman, EE Dept., University of North Dakota, PO Box 7165, Grand Forks, ND 58202; (701) 777-4331 Compmail+ j.hootman

From the Editor-in-Chief

The Far East Issue for 1990

Last month I discussed the fact that about one third of the readers of *IEEE Micro* live outside the United States. This issue concentrates on one of those areas. It shares information about some of what is going on in Japan in microprocessor-related areas. Ken Sakamura has done an outstanding job of obtaining papers and guest editing this issue.

It is important that every engineer be exposed to the work and thinking of an international nature because their very futures may depend on how they and their companies view themselves in the international marketplace. As communications between countries become easier and as the world becomes more open, it is critical that we all develop a broader international view of products and our work.

Recently, I've read with a great deal of concern about the closure of U.S. Memories, the decrease in profits for IBM and Apple, and the downsizing of many of the high technology companies. With the clear indication that the US military budget will decrease in the long term, manufacturers of high technology in this county will have to find some new ways to survive and develop products in the very competitive commercial market.

I informally surveyed the advertising carried in *Forbes* (February 19, 1990) and in *Business Week* (January 19, 1990) to see how US-based industry is doing in selling to its own market. To my surprise, I found about 21 pages of international advertising in *Business Week* that were selling technical products and only 10 pages of advertising that sold technical products manufactured in the US. In *Forbes* the numbers shifted toward the US. I counted 25 advertising pages on US technical products vs. 19 pages of international advertising. It did not surprise me that the vast majority of advertising in both publications was of a service nature.

It seems we would see more advertising for American-made products if they were available and competitively marketed. Perhaps US manufacturers have been taking too short a view about the way products are conceived, developed, and marketed so that the products will be available and refined when the demand is there for them.

For example, one of the articles in this issue discusses the development of a 4-bit Josephson technology computer, a project that was dropped by IBM (the only US company exploring this device) several years ago. Hitachi management must either see a future for this type of technology or feel they will learn some things over a period of time that will aid them in the development of future products. Perhaps this example reflects one of the major differences in philosophy between the US and other countries. US manufacturers need to look around and learn.

(The most recent readers' suggestions appear on the next page.)

Become a reviewer and join the #1 team.

Want to help keep *IEEE Micro* #1? Editor-in-Chief Joe Hootman seeks more technical reviewers—people interested in seeing that the articles published in *IEEE Micro* continue to be of the highest quality. The technical review process is a crucial step in providing readers with correct and timely information so they can keep up with their ever-changing profession.

Send your professional information directly to:
Joe Hootman
University of North Dakota
PO Box 7165
Grand Forks, ND 58202

In the Mailbag

June 1989

"I would like to see more articles on microprocessors designed for a concurrent environment." J.H., Vienna, VA

"I liked [the] announcements and New Products. I would like to see more information and products." R.S., Alexandria, Egypt

"I [dislike] the fact that you edit a special Far East issue." A.C.V., Valparaiso, Chile (See my comments in this column.— J.H.)

August 1989

"I would like to see information about RISC architecture and its fundamental concepts." M.R.S., Tehran

"I liked the thorough and well-written articles from Intel Corp. and AT&T Bell Labs as well as the Micro Law series. I disliked the 'salesman' style in the articles from Analog Devices and Motorola Inc. I would like to see a thorough review and comparison of microprocessor architectures (and philosophy) and more covering of microcontrollers." E.A.B.B., Caracas, Venezuela (We try to eliminate, insofar as possible, the blatant sales information in all articles. It is not possible to take all mention of companies and company products out of an article because that is what is being discussed in the article. How much 'salesman' type of information appears in a given article is largely a subjective call of the reviewers.—J.H.)

"I liked all of the articles [on] the new-generation microprocessors. I would like to see more information on how to get manuals or books about company products." J.V., Lomas DeZamora, Argentina.

"I would like to see more about hardware of IBM-PC computers and also about networking PCs." M.R.P., Tehran

"I liked the Intel i860 microprocessor and MC68332 microprocessor [articles]. Would like to see more articles like these." A.J., Bombay, India

"I liked [the] information on the new Motorola MC68332 microcontroller—clear and professional description of this new product." V.S., Vancouver, Canada

October 1989

"I liked Intel's job advertising on page 1!! Micro View—CISC and RISC ideas will become merged, and future distinctions will be difficult to make. What do you expect when you have 10 million transistors to play with?" J.D.T., Tokyo *(IEEE Micro* has never been a magazine that has attracted advertising, and the Intel ad is very welcome. I do not understand why *Micro* does not get more job ads.—J.H.)

"I liked to read about and [would like to] receive more [information] about 'Microprocessor Communication Using Dual-Ported RAMs,' Especially the Am2130 DPR." S.N.B., Tehran (You may contact the authors directly at the address indicated after their biographies at the end of this article.—J.H.)

"I liked [the] information about using dual-ported RAMs for interprocessor interconnections. I would like to see more algorithms [used for] digital signal processing." M.H., Tehran

December 1989

"I loved the entire issue. I would like to see more of the same." T.H.W., Long Beach, CA

"I liked SCSI news in New Products." D.F., Fort Wayne, IN

Letters

On second thought...

To the Editor:

I greatly enjoy reading Richard Stern's Micro Law column in *IEEE Micro*. When he first began his series on protection of screen displays, I wasn't sure how I felt about it. But now that I've had a chance to think about things in the context of my own experience in the industry, I thought you might find my comments relevant.

I have worked as a software engineer in companies ranging in size from one employee to 1,000 and more. Each product I've worked on had some unique features that made it different from anything else I was aware of. Hopefully, these differences would set the product apart from others in the marketplace enough so it would be successful and profitable.

A good user interface has always been important, and some of these products counted that area as one of their uniquely superior features. However, no company I have ever worked with has ever considered protection of the user interface as an incentive in producing the product. Furthermore, I have come to the conclusion that there are no reasonable circumstances under which *any* company would consider the availability of protection for screen displays or key sequences as a significant factor in the decision to go ahead with a software project.

The reasons for this are fairly simple. Suppose, for example, that we're talking about a user interface that is significantly advanced over any existing competition. Should a company go to the expense of developing the product when competitors are allowed to copy its operation? The fact is, it takes several person-years to develop a modern user interface, even with a solid specification of what it should do. If the new user interface really is a major step, its developer can count on a substantial time lag before it must face imitators—unless those ideas are already being developed by the competition.

A user interface that does not break new ground only needs protection to stop cloning. But clones are only produced when a product becomes so popular its user interface becomes the standard. Any product facing this prospect certainly does not need user interface protection as an incentive. It's unlikely anyone in the boardroom of Lotus Development in 1982 said, "We won't come out with 1-2-3 unless we can prevent product imitations a few years from now when we have 70 percent of the market."

When a product does break new ground, it is the ideas that are important, not the words and images. Suppose a database manager program has a new way to perform a "join" between two databases. Instead of using a procedural language like SQL, the user only needs a mouse to "grab" a field name in one database, and "drag" it over to the joining field name of the other. This is a very conceptual operation, independent of the detailed words and graphics on the screen. Are we going to allow a copyright of this idea? It seems like any really new idea should be considered for patent protection—and require the usual evidence of its novelty.

The microcomputer industry has thrived more because of imitators than despite them. And the public has done nothing but benefit. I have personally been involved in both sides of cloning. As the original author of MS-DOS, I have been told by Gary Kildall, founder of Digital Research, that I "ripped off" DR's CP/M operating system in developing DOS. Since then, Digital Research has made a major business out of its DOS clone, DR DOS. Several other DOS clones or substitutes exist, and Microsoft has not accused them of violating any protections on DOS. These products were all developed independently, meeting DOS's specifications—that is, imitating its ideas—but not stealing the code itself.

That is the protection that software developers want and need. Anybody can have ideas, or copy somebody else's. But writing the code and making it work is what it's all about. Microsoft hasn't tried to stop DOS cloners, but they have seized over 75,000 copies of real MS-DOS shipped to the US by software pirates in Taiwan. Copyright protection is only needed to protect the code itself, not what it does.

Tim Paterson
Renton, Wash.

To the Editor:

I would like to highlight one point that has been missed by the authors of "A Comparison of RISC Architectures," *IEEE Micro*, August 1989. [*The authors are R.S. Piepho and W.S. Wu.—Ed.*]

The relative addressing capability of a given architecture allows one to create position independent code, that is, code (and data) that can be mapped anywhere in memory.

This capability is required to create shareable libraries, a feature that is available in the SunOS 4 and Unix System 5.4. Shareable libraries will be more and more needed as system and user software becomes more modular and layered. Thus a 16-bit relative addressing capability limits the size of a shareable library to 64 K.

J. Luu
Igny, France

About the Cover

The TRON Intelligent House

Ken Sakamura
University of Tokyo

The cover of this issue depicts the TRON Intelligent House built in the Roppongi district of Tokyo. Approximately 1,000 computers interact to support this house. The direct cost of building on this 300-square-meter lot was 1 billion yen (US$6.7 million), and the additional indirect costs reached 2 billion yen (US$13.4 million).

The TRON Intelligent House is the experimental prototype of the future house, which incorporates the results of the TRON Project that proceeds under my leadership. The house reflects my idea of the house of the future. Many parts of the TRON House have sensors, computers, and actuators. These network-connected devices can work with each other in a cooperative manner. For example, if the controls of two subsystems seek conflicting objectives, an overall control instructs them to compromise.

I based my concept of the intelligent house on the idea of the "intelligent object" that uses computers to control its behavior. I designed the house, however, with an express purpose of not placing computers in visible places and using natural materials and natural surroundings to best advantage to create a pleasant environment. Computers help run the house to maintain comfortable living conditions only when problems arise.

With this house, the air conditioner does not turn on automatically just because the weather becomes hot. If the sensor and the controlling computer reflect that the natural air circulation caused by external winds is just right, the computer opens the windows. Only when the use of the air conditioner is deemed appropriate (for example, in case of rain) does the computer turn on the air conditioner.

Cooperation among these intelligent objects is very important. A few examples may illustrate this point. If we want to use a window blind to control illumination, we can build an automatic blind that monitors the amount of direct sunlight in a room. We can't just use an automatic blind with a brightness sensor for this purpose as we probably don't want the blind to shut out the sunlight completely. So, unless we program the blind to control the amount of sunlight it lets in—just as people do when they handle blinds—the automated house will have rooms that darken suddenly in broad daylight. To intelligently control the level of a room's brightness, many sensors become necessary—including one that monitors the position of the sun.

On top of this, people probably won't want to see a blind close when they are looking outside a window. To avoid such occurrences, the computer must monitor people's positions in the room. We may even want to suppress the automatic closing of the blind when people remain near the window over a certain amount of time. If we also have intelligent lamps that are programmed to keep the brightness constant inside a room, the closing of an automatic blind may next turn the lamps on—even if it is bright enough outside. Worse, we may see some kind of system oscillation between these intelligent objects.

My brightness example shows that the ad-hoc approach to control human habitats can end up with a very bad performance/cost figure. Instead, we need to consider seriously all possibilities. Sensors to monitor human movement can be part of security devices. Brightness sensors can be shared by blinds, lamps, or solar energy devices. We should be able to cut down overlapping costs and must avoid unnecessary interferences among these intelligent objects.

When we look at home appliances today, we notice that many of them al-

0272-1732/90/0400-0006$1.00 © 1990 IEEE

ready use computers. The inclusion of computers certainly has merit. However, demerits exist, too. The major problem here is that each appliance seems to have been designed with the view that it will be operated in isolation. In reality, this seldom happens. When we activate a number of these appliances, each with its own sensors and reaction mechanisms, we cannot forecast exactly how the total system will behave. Waste of energy and natural resources as well as unexpected accidents can be a reality. What is more, the lawsuits arising from such unfortunate events could be a future problem.

So, I stress here that the computerized appliances of the future must communicate with each other to offer good service to users.

Why, then, don't today's appliance designers take this total approach? Generally, it's because designers know consumers will select many appliances from various manufacturers and will always purchase additional devices later on. As a result, the operation methods in appliances have no standard interface. With this background, no manufacturer will be able to take a system approach to appliance design unless supporting infrastructures such as standardization are available. The TRON Project aims at offering such infrastructure.

Let me explain the functions of the TRON House briefly. The objective of this pilot house is to allow designers to participate in many experimental approaches rather than offer a practical house immediately.

The TRON House includes housewide networks and sensor, telephone, video, audio, and utility subsystem networks. A central computer system placed in the basement, and called the house server, monitors and controls these networks and handles the database.

The sensor network keeps track of temperature, humidity, wind direction, air pressure, carbon dioxide content, rain, interior air circulation, and people's positions. The multifunction telephone system contains a digital PBX that controls the telephone, intercom, and computer communication. A video system broadcasts the images taken by internal cameras of the entrance and living rooms, the data from each subsystem, and television data received from outside. An audio system feeds sounds to the listeners from internal sources, external radio sources, and combinations of these.

A system that consists of sensors, automatic windows, four heater/cooler systems, a water spray machine on the roof, and fans to draw in the external air controls the air conditioning.

Computers control the lighting so that it allows different lighting scenarios for each room. This system also controls the closing and opening of curtains. The kitchen has a computer cooking guide that uses the multimedia functionality of the BTRON computer (another TRON Project feature). This computer controls the cooking utensils and even dispenses the right amount of seasonings as needed.

Your health can be monitored by a computerized toilet facility that performs a urinalysis automatically. An automated storage system can place unused objects into containers and move them to the basement warehouse. An automated garden system grows plants using hydroponic techniques, and water-level sensors control the water supply. The security system is based on external infrared sensors and internal heat sensors.

A computerized house is useless if the people living there can't make use of it easily. So, operation of all the switches in the house is designed according to the TRON Design Guideline. Today, we find that the standardization of operation on computer workstations attracts much attention. What we have done is apply the same idea to the switches in our surroundings. Inhabitants in the house can control operations with standardized switches, intelligent remote controls that have receivers in every room, or BTRON computers.

Over the course of three years, we plan to let people live in the TRON House so that we can monitor the data and analyze the cooperative behavior of intelligent objects. We will also try to gather valuable experience from simulated computer breakdowns, fires, and intruders.

Micro Law

Richard H. Stern
Law Offices of Richard H. Stern
1300 19th Street NW, Suite 300
Washington, DC 20036

Software patents

Concern has been expressed in the computer industry about software patents. Articles appeared in the *New York Times* last year, and last spring the Massachusetts Institute of Technology held a forum entitled "Software patents: A horrible mistake?" What is the basis of the concern?

Programmers fear that patents will be granted that monopolize basic concepts (such as pull-down windows or the use of menu bars) or algorithms that programmers need to use to write effective software. The Patent and Trademark Office seems to be willing to issue virtually any software patent presented to it, and it lacks facilities to search out the prior art in the field. Therefore, very soon we may see patents drifting around like derelict ships or unseen icebergs, with which unsuspecting software entrepreneurs may disastrously collide.

The problem is worsened by the fact that patent applications stay on file in the Patent and Trademark Office in secret for about three years before patents issue on them and are published. So businesses may be established on products that suddenly become infringing when a patent springs into existence.

The concerns are legitimate, but they are somewhat exaggerated. Some of the problems can be solved by moderate institutional changes. Others, however, may resist solution.

Consider the invention or discovery of a truly novel and technologically advanced software technique, unobvious to other workers in the field at the time. As of early 1990, it is unclear whether the inventor can secure patent protection on the technique. Illustrative techniques, for the purposes of this discussion, are

- the use of windows (which may be considered as having two programs running with both displays shown on parts of the same screen);
- the fast Fourier transform (FFT);
- a new and much faster algorithm for linear programming or sorting (such as that described in AT&T's recent patent on Narenda K. Karmarkar's discovery of a new method for linear programming); and
- discoveries or concepts such as $E = mc^2$ or Thevenin's theorem.

First, one must distinguish between the patents issued by the Patent and Trademark Office and the patents upheld by the courts as valid. Sometimes, now approximately a 20-percent overall average, the Office issues patents that are not valid. They may be invalid either because of a mistake or more often because the Office did not know about the anticipatory work of others in the field. The 20-percent figure may be exceeded in cutting-edge technology, in which the Office may be less informed.

(As an aside, the validity/invalidity ratio used to be 1:1 about a decade ago. But early in the 1980s Congress established a special appeals court, the US Court of Appeals for the Federal Circuit, or CAFC. This court hears all patent appeals and brings about greater uniformity of decision in patent law. Since that time, the court has upheld relatively more patents as valid—now approximately 80 percent instead of 50 percent. Observers generally consider that the new court is more favorably disposed toward intellectual property rights than the regional courts of appeals were in the past. This disposition may at least in part account for the changed ratio.)

In the case of software, the problem of a perhaps insufficiently informed Patent and Trademark Office is particularly acute. It is harder to locate well-indexed textbooks for searching out prior-art software than for prior-art electronic circuitry, mechanical devices, civil or chemical engineering, and other older fields of art. *Chemical Abstracts* and similar publications index the prior work in many fields and make it easy to find. But software prior art is typically difficult to find. Often, it exists only in instruction manuals in someone's desk drawer or even only in disks or tapes.

At this time, no indexing or information retrieval system for software exists that is comparable to that for other fields of art. Perhaps, this defect will be remedied in time. But until then, the quality of review of prior art in the Patent and Trademark Office before software patents are granted may not be up to the level of quality for chemicals, drugs, circuitry, process technology, and machines. Moreover, it is unclear whether the personnel of the Office who examine software patent applications are as skilled as those who examine other

0272-1732/90/0400-0008$01.00© 1990 IEEE

arts. These factors converge, at least at the present, to make software patents less reliable than other patents.

A second problem is the lack of clear legal definition of patentability in this field. Part of this problem centers on an apparent change in decisional law. Part of it results from a possible discrepancy between what the Patent and Trademark Office does and what the Supreme Court has said. The Supreme Court has decided software patentability four times, each time in the context of a review of a decision of the Patent and Trademark Office's denial of a patent to an applicant.

In the first case, the Benson case, in the early 1970s, the Court held unpatentable an algorithm for converting BCD, or binary coded decimal, numbers to pure binary. The Court ruled that the formulation of the supposed invention in the claims of the proposed patent was too abstract or sweeping. For all practical purposes the claims covered any use of the algorithm in a general-purpose digital computer, so that the effect would be to give a patent on the algorithm itself.

Such highly abstract formulations of ideas could not be patented, the Supreme Court held, because to patent them would hinder, more than further, the progress of useful arts. Algorithms and general laws such as $E = mc^2$ are the tools of technological workers, essential to technological progress. It would be contrary to patent policy to deny the use of such tools to everyone but the inventor, even if the inventor first discovered them.

In the second case, the Johnson case, the Supreme Court held the claimed invention to be obvious. It did not pass on the question of whether the subject matter was inherently unpatentable as an abstract idea.

In the third case, the 1978 Flook decision, the Court addressed an "alarm" system for chemical processing. The system calculated a moving time average (by smoothing data by an exponential factor). When the current value exceeded the value of the moving time average by more than a predetermined value, an alarm was to be sounded. The Court considered the system to be a mere numerical calculation, rather than a true mechanical system, since the overwhelming majority of the parts of the so-called system just performed numerical calculations. This decision placed the case on a par with Benson; that is, the patent effectively sought to claim the algorithm or formula itself.

In so ruling, the Court assumed that the smoothing algorithm was new (although it was not). Then it said that in assaying patentability any law of mathematics or physical principle or other law of nature, even if newly discovered, had to be regarded as if it were already in the public domain. Otherwise the result would be to grant patents on laws of nature such as $E = mc^2$. Therefore, a system using a natural law could be patented only if the inventor's machine-implementation of the law was novel and unobvious.

Let's assume patentability is a 1, and unpatentability is a 0. Then, we have a case of the patentability value of the natural law being added to the patentability value of the particular implementation into machine form that the inventor describes. The patentability value of the natural law is always 0. So to secure a patent, we must agree that the patentability value of the inventor's machine-implementation must be a 1, not just another 0, for $0 + 0 = 0$. In effect, that is what occurred in the Benson and Flook cases.

(Readers will have to assume a certain amount of bias in my description of these three cases, because I was counsel for the government. Others may interpret the issues and results differently.)

In the fourth case, the 1981 Diehr case, a 5-4 Supreme Court reversed the Patent and Trademark Office's refusal to issue a patent on a rubber-molding system. The system used temperature sensors to measure the temperature of rubber curing in a mold, and then it established an integral of heat over time. When the integral reached a predetermined value, the mold was opened.

The formula for the integral (an exponential function of temperature and other variables) was old. But apparently government counsel did not show the apparatus with embedded temperature sensors to be known or obvious. The Court said that while an algorithm for numerical calculation was unpatentable, the system before it was really a new machine system that used an algorithm (or formula) to govern its operation. The Justice who wrote the Court's opinion in the Flook case dissented, on the ground that the facts of Diehr were not distinguishable in any material way from those of Flook. Since this 5-4 decision, the Supreme Court has not addressed software patents.

The Supreme Court decisions appear to stand for a rule that algorithms (and thus programs based on them) can be protected by patents only when they are claimed in the context of an overall machine system incidentally using the algorithm. The mechanical parts of the system must not be so trivial that the system does little more than calculate numbers; there must be something more than just the algorithm. Also the machine system should be an unobvious, or at least novel, way to use the algorithm. Whether the algorithm is new or old would not appear to matter. However, as a matter of common sense, it stands to reason that an implementation of a new and unknown algorithm will be more likely to be novel than will be an implementation of an old, well-known algorithm.

Since the Diehr decision, the law in the lower federal courts has become rather confusing. In the Merrill Lynch case several years ago, a federal trial court upheld a patent on what seems to be nothing more than a method of calculating numbers and switching money from one account to another in accordance with the result of the calculation.

Very recently, the CAFC upheld one Patent and Trademark Office denial of a patent and reversed another on seemingly identical ways of claiming inventions involving algorithms. However, in one of the cases the form of the patent claims was a process for doing something that looked like a numerical calculation. (There the CAFC upheld the Office's refusal to grant a patent). In the other case the form of the patent claims was a system for doing something that looked much like a numerical calculation. (Here the CAFC reversed the Office's refusal to grant a patent.)

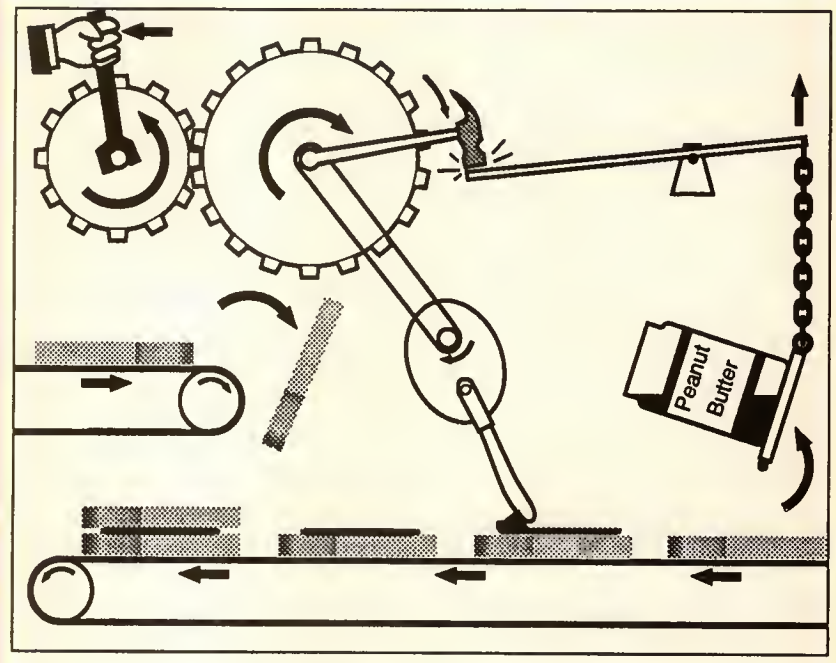
There may have been some more apparatus-like trapping in the latter claim set (for example, a ROM was men-

tioned). But experienced patent lawyers can always translate any claim for a series of steps (a process) into a claim for a system indistinguishable in substance from the process/steps claim. That is, counsel in the latter case claimed that the patented system was a set of "means" for doing what the corresponding steps would do. It was a system made up of an interconnected set of "means." (See accompanying box.) This pair of decisions leaves patent lawyers in the software field quite perplexed.

In the November 1989 Grams case, the applicant claimed a method of diagnosing an abnormal condition for a patient having a number of test parameters from lab tests. The parameters are compared with "normal" reference values. The Patent and Trademark Office and CAFC considered this procedure to be the same as what a doctor does in the ordinary course of business. The sole physical step in the procedure was to obtain test data; everything else was manipulation of the data.

In the Iwahashi case decided four days later, the CAFC reversed the Patent and Trademark Office for refusing to allow a patent on an autocorrelation unit for recognition of patterns, such as voice patterns. The ordinary method of calculating correlation used a multiplication process to calculate a product of signals, but that method was unduly expensive in terms of computer utilization. The Patent and Trademark Office said that the difference between a series of steps and a set of interconnected means was nonexistent in this context. But the CAFC said that this was incorrect, because the claimed means extended only to those identified in the patent plus the equivalents of those so identified.

Meanwhile, the Patent and Trademark Office has issued a large number of software patents, after holding them for a very long time. Many of the newly issued patents seem to be on systems of doing business or calculating, without much in the way of specific apparatus details. Will the courts uphold these patents? At any rate, the patents may pose a threat to software that others developed and marketed during the several-year period in which the applications for these patents were pending in the Office.



How to Change a Process into a System

Consider the following claims for a method of and system for making a peanut-butter sandwich.

Claim 1. A method of making a peanut-butter sandwich comprising the steps of:

1) placing on a table-surface a first slice of bread having upper and lower surfaces, said lower surface of said first slice of bread being adjacent to and above said table-surface;
2) spreading peanut butter on said upper surface of said slice of bread, thereby covering said upper surface with a layer of said peanut butter; and
3) placing a second slice of bread over said layer of peanut butter and generally contiguous therewith.

Claim 2. A peanut-butter-sandwich-making system comprising:

1) support means for supporting bread slices;
2) means for transporting an initial bread slice to said support means whereby said initial bread slice is moved onto and is supported by said support means;
3) means for dispensing peanut butter onto said initial bread slice supported by said support means and for spreading a layer of said peanut butter over said initial bread slice; and
4) means for transporting a further bread slice to the vicinity of said support means and for placing said further bread slices over said initial bread slice and generally contiguous therewith.

These developments have led to much of the concern described in the first paragraph. To some extent, these developments could be remedied by modest institutional changes, some of which have already been proposed. For example, the Patent and Trademark Office could establish procedures under which it published pending patent applications after a certain period, as occurs in Europe. To do this would require an amendment to the patent laws. Additionally, funds could be appropriated (if they could be found) to devise a better system for finding software prior art. Each of these measures would improve

the present situation. Perhaps, in addition, the Patent and Trademark Office could also hire more skilled software-examining personnel.

Whether the law can be made more clear by judicial evolution of precedent is more open to doubt. The legal test is not all that clear, and it may rest on some highly abstract or metaphysical distinctions. Further, the lower federal courts appear to have ideas different from those of the Supreme Court.

Even more fundamentally, an inherent problem exists with the application of the patent system to software. The US patent system, unlike those of many other countries, provides for a more-or-less absolute monopoly. A patent entitles an owner to refuse to license its patent, or to charge as high a price for a license as it chooses. Except for public health and safety situations, this rule is nearly absolute. (An example of the exception is a 1934 case. There, Milwaukee would have had to dump sewage into Lake Michigan if enjoined from committing infringement of a patented sewage treatment process.)

From one standpoint, algorithms, FFTs, discoveries like $E = mc^2$, and the like, deserve the incentives and rewards of the patent system just as much as any other advances. There may be little incentive for anyone to spend time, money, and effort on developing mathematical algorithms for free. We would promote the progress of science and useful arts by encouraging more advances of that sort.

Yet, the prospect of holding such rights away from others, at the patent owner's option, for 17 years, or authorizing the patent owner to allow their use only at a very high price, may suggest that such patents cause more harm than good. That is, in effect, what the Supreme Court said in the Benson and Flook cases. Probably, we should be thankful that no one patented the fast Fourier integral. Such a patent would probably more hinder than promote the progress of science and useful arts.

The intermediate, compromising, maximizing solution would let us say that others could use these advances during the life of the patent, for a reasonable fee. For example, suppose someone sells disks in which an encoded computer program implements the FFT algorithm. What is wrong with a system under which the seller of the disks pays 25 cents per disk to the creator of the algorithm? Fairer than a free ride, such a plan would provide incentives to algorithm development. Also it would not create the obstacles to others' research and development programs and to technological progress that the Supreme Court felt calls for refusal to allow ordinary patent protections for algorithms.

What would be wrong with a system of paying whoever created bubble sort 10 cents for each sort performed during the first 10 years after the first publication of bubble sort? We would not see 75 years of injunctions, confiscation of infringing products, and jailing of infringers, as prescribed by present copyright law—just modest compensation. That kind of system would be better than either 1) excessive legal protection under patents or copyrights, or 2) inadequate protections under a free-ride system.

Those with existing patent rights would violently oppose the first step toward compulsory licensing of all patents.

But the present US patent laws—or for that matter the present US copyright laws—do not contain that option. In fact, the option would probably violate the Berne Copyright Convention, a copyright treaty to which the US just became a signatory. Probably, it would not be possible to make this kind of change to US patent (or copyright) law. Those with existing patent rights would violently oppose the first step toward the slippery slope of compulsory licensing of all patents. Rather than suffer that change, they would rather have no patents for algorithms, $E = mc^2$, and the like. Thus, the compromise solution is not feasible under the patent law and copyright law rubrics.

What that means is that if things of this type are to be protected, we would have to set up some other kind of system. For example, the Semiconductor Chip Protection Act of 1984 legally protects chip layouts in a limited way for 10 years. The same concept might appropriately be extended to algorithms, and perhaps to languages and instruction sets, which most observers agree cannot be protected under the patent or copyright laws. Perhaps the concept could also be extended to various aspects of user interfaces.

To be sure, you'd have to take the idea to Congress. But if you can take to Congress the idea of amending the anti-trust laws to facilitate HDTV development or to allow joint manufacturing consortia, you can surely do the same for algorithms. Algorithms are probably even more socially important and useful to encourage.

In the meantime, it is unrealistic to anticipate much security of expectation in the field of software patents.

The Current Japanese Computer Scene

Ken Sakamura

University of Tokyo

This is the fourth special issue on the Japanese computer scene that *IEEE Micro* has brought to its readers. In the last four years many things have happened in the world semiconductor and computer industry. I'm always amazed at the speed of the progress in microelectronics. Thanks to this technology, information can travel around the globe ever faster and more easily.

The warming of relations between the West and the Eastern Bloc countries owes a lot to this technology. However, many of us still find it difficult to discover what is going on in the minds of other nations' peoples. In this sense, the special issues of *IEEE Micro* on European computing and Far Eastern computing are very valuable to readers. They present information that helps readers understand the rest of the world to which they don't ordinarily pay much attention.

Before going into the introduction of this issue's articles, let me summarize the latest status of the TRON project on which I've reported in past issues. The project under my leadership now includes more than 140 companies from Japan, the United States, and Europe. The project envisions a future in which electronic appliances become intelligent objects. Such objects contain microcomputers, sensors, and actuators to offer "intelligent" service. The intelligent objects will be connected by a network so they can offer sophisticated services. A Highly Functionally Distributed System (HFDS), which is the major goal of the TRON project, will connect 10 thousand, a million, or even a billion such intelligent objects and operate as a cooperative distributed system. The TRON project tries to design the necessary technology to build such an HFDS.

ITRON, which is an embedded operating-system kernel specification to be used for intelligent objects, has seen diversification. The original version, now called ITRON1, has been joined by a simplified version called micro-ITRON that runs on an 8-bit computer. ITRON2 targets 32-bit CPUs, including the TRON VLSI CPU. ITRON2 contains rendezvous, message buffer, local memory pool, forced exception, and resource management functions on top of ITRON1's functionality.

BTRON, an operating system specification for personal computers, will support controllers of intelligent objects that are hooked together via networks. BTRON designers aim to make the system compact and low cost.

0272-1732/90/0400-0012$01.00 © 1990 IEEE

BTRON1, a simplified version that runs on the iAPX286, is available from Matsushita Electric Industrial Co., Ltd. now; BTRON2, which targets the TRON VLSI CPU, nears completion of the initial prototyping.

The TRON VLSI CPU specification, which can be used as a general-purpose CPU and as the core of the intelligent objects, now has a 64-bit specification in addition to the 32-bit specification. Implementation of the CPU is now available from the Gmicro group (Fujitsu, Hitachi, and Mitsubishi) as the Gmicro/100, Gmicro/200, and Gmicro/300. Gmicro/300 is a 17-MIPS CPU. Toshiba is now implementing the TX1 and TX3 versions of the CPU, and Matsushita has finished a prototype implementation. Oki Electric works on its own implementation. The Gmicro group plans to develop a CPU that is five times as powerful as the Gmicro/300. Software support tools for these chips are available from US companies. Microtec Research Inc. and Green Hills Software Inc., for example, supply compilers, while the Software Component Group and Ready Systems produce real-time operating systems.

Another application of the TRON project, an experimental future house, has been built. (Please see About the Cover on page 6 for further information on the TRON House.) Additional applications include the design of office buildings, automobiles, and urban areas to offer the infrastructure of a future computerized society.

This special issue contains original articles on VLSI research and commercial VLSI products from Japan. People outside Japan may think that the Japanese semiconductor makers fabricate only dynamic memory chips. Of course, we have researchers designing very unique devices.

Mitsubishi, Matsushita, and Hitachi contribute research reports. Mitsubishi authors discuss a 1M-bit DRAM chip with a built-in, 8-Kbit cache memory circuit. Such projects will make the design of no-wait-state memory systems very easy.

The Matsushita article presents a processor element for the parallel computer called ADENA. ADENA uses 256 processors for scientific calculations; the interconnecting network is its unique feature. With this computer, performance comparable to a pipelined supercomputer can be achieved at a much lower cost. The processing element's ALU, a single VLSI chip, uses a 64-bit-wide bus and achieves a peak performance of 20 Mflops.

Hitachi authors tell about a very fast, 4-bit microprocessor that they built using a Josephson junction device. This processor uses about 2,000 gates (based on the Josephson junction) and has a clock cycle rate of 1 gigahertz. The authors discuss the possibility of building practical computers using the Josephson junction device in this decade. In Japan, Fujitsu and the Electro-Technical Laboratory of the MITI government agency have announced similar prototypes. It is true that the prototypes are not practical computers yet. However, the day may not be far away when we will see usable computers with Josephson devices that run faster than gallium arsenide devices.

Two commercial VLSI products, both 32-bit CISC processors, have been designed by NEC and Fujitsu. Both CPUs are comparable to Intel's 80486 and Motorola's 68040 and perform at 10 to 20 MIPS. The V80 CPU from NEC is an upgrade of its own V70 CPU and runs four times faster. NEC authors discuss how they achieved this performance gain. Fujitsu's Gmicro/300 CPU is based on the TRON VLSI CPU specification. The design goal was to execute register-to-memory operation in one machine cycle. This article will be published in the next issue, due to page limitations.

I hope the articles presented here will help you to gain knowledge of Japanese computer technology. Such understanding of each nation's technology will go a long way to resolve politically charged trade issues. If this special issue gives you any valuable information, I will be delighted.

As has been the case on previous occasions, I would like to thank the fine editorial staff of *IEEE Micro* for their help in producing readable English articles.

Ken Sakamura is an associate professor in the Department of Information Science at the University of Tokyo. He initiated the TRON project in 1984. Under his leadership, several universities and over 100 manufacturers now participate in the project to help build computers in the 1990s. In addition to his involvement with TRON, he chairs several committees of the Japan Electronics Industry Development Association and the Information Processing Society of Japan.

Sakamura received the BS, ME, and PhD degrees in electrical engineering from Keio University in Yokohama. He has written numerous technical papers and books and is a member of the IEEE and the IEEE Computer Society.

Questions concerning this article can be directed to Ken Sakamura, Department of Information Science, Faculty of Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan.

FEATURE

The Cache DRAM Architecture:
A DRAM with an On-Chip Cache Memory

We fabricated the Cache DRAM, a hierarchical RAM containing a 1-Mbit DRAM for main memory and an 8-Kbit SRAM for cache memory, using a 1.2-μm CMOS technology. Suitable for no-wait-state memory access in low-end workstations and personal computers, the chip also serves high-end systems as a secondary cache scheme.

Hideto Hidaka
Yoshio Matsuda
Mikio Asakura
Kazuyasu Fujishima

Mitsubishi Electric Corporation

Since the invention of the 1-Kbit, MOS (metal-oxide semiconductor) dynamic RAM some 20 years ago,[1] DRAMs have successfully and consistently provided the main memory in many generations of computer systems. The MOS DRAM is, in principle, suitable for low-cost, high-capacity random access memory. Thus it found its own niche as the main memory in a wide range of computer systems and in various peripherals. Furthermore, in each generation DRAMs have satisfied requirements posed by computer systems that demand bit-cost reduction, high capacity, and improved performance.

The DRAM's success comes mainly from industry attempts to scale down the physical dimensions of large-scale integration (LSI) and other technological breakthroughs. Such innovations include the 4-Kbit, one-transistor, one-capacitor memory cell; 16-Kbit NMOS dynamic circuitry, 64-Kbit CMOS implementation; and 5V-only operation.

Especially in low-end personal computers, DRAMs achieve the access speed that microprocessor units require. However in recent years we've seen MPU clock rates rise higher than did those in DRAMs. The difference between the DRAM access speed and the MPU's requirement becomes a limiting factor in system performance.

Even if the same CMOS fabrication process is used, MPUs enjoy a larger benefit from scaled-down devices than do DRAMs. Memory chips benefit less because the memory chip's access time is not dominated by pure logic operations but by such factors as cell signal-sensing operation and word-line delay. From this background we see that memory access with wait states in combination with internal pipeline operations must be implemented in recent MPUs. This scheme however has drawbacks in the complexity of control and the degradation of the total system performance.

Figure 1 depicts an example of this "speed gap" between MPUs and DRAMs. If we assume a memory access with no wait states for a CPU clock rate of 33 MHz, DRAM access would take no more than 30 nanoseconds. This speed is beyond the reach of present-day CMOS DRAM technology. Even for one-wait-state access, the present 4-Mbit DRAM generation can barely provide the required time of 60 ns. As is well known, this situation is more serious in the recent high-performance RISC (reduced instruction-set computing) types of MPU systems. Clearly the present DRAM does not keep pace with MPU speeds, and in the near future DRAMs will lose their role as the main memory devices in these systems.

0272-1732/90/0400-0014$01.00 © 1990 IEEE

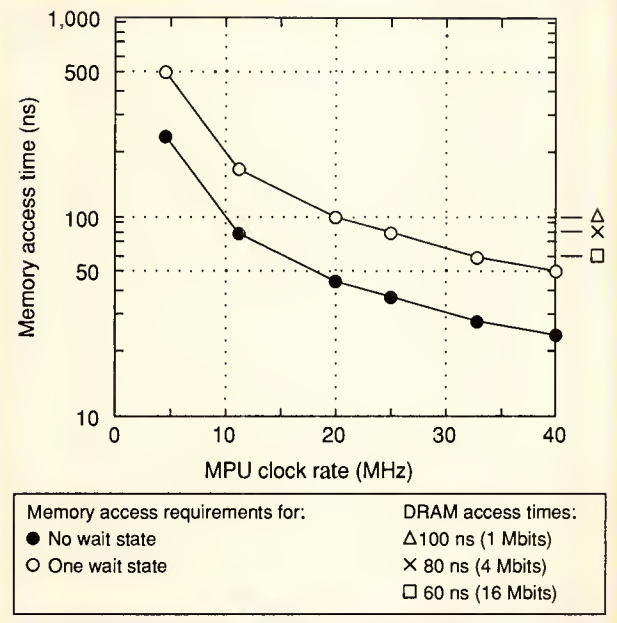


Figure 1. The speed gap between DRAMs and microprocessors.

In MPU systems, a DRAM's access speed often limits the total system performance. A bottleneck in today's advanced MPU systems, slow memory access sheds its shadows on the whole spectrum of the microprocessor-oriented systems. Solutions must be explored to enhance system performance and also to make the most of the original advantages of DRAM memory cores.

In high-end systems, some measures cope with this speed gap. Cache memory implementation is a common solution, one that also finds favor in MPU systems. But in cost-sensitive low-end systems, the slow memory access inhibits reduced-cost performance improvements. Watanabe et al.[2] reported a whole cache memory system implemented in one chip, and others reported a number of cache memory implementations on MPU chips. These solutions tend to boost system cost and are not effective in low-end systems.

On the other hand, we've heard of very large scale integration (VLSI) DRAMs, 4-Mbit devices with a 0.8-μm design rule soon to be introduced into the market. Also 16-Mbit DRAMs are under development for next-generation main memories. As for low-end systems, we foresee that a single DRAM chip will cover the whole main memory capacity in the near future. The configurations of main memory systems should be reconsidered from this viewpoint, for cost optimization.

Conventional standard DRAM design approaches don't serve this area of applications effectively. The interface and some additional functions in the memory system such as error checking and correction (ECC) and DRAM refresh control can be favorably incorporated on the DRAM chip. These integrations would cut the signal delay on the interchip wiring.

Historically in the standard DRAM design, the measures taken for performance-related requirements have concentrated almost solely on scaling down the device size and improving the fabrication process. But the present situation suggests some other approaches in certain application areas. Present-day environments of VLSI memory technology offer many opportunities for system optimization based on the architectural evolution of DRAM design.

Table 1.
Alternative measures to bridge the speed gap.

Choices	Performance	Cost
DRAM interleaving	Poor	Low
DRAM page mode/ static column mode	Good	Low
Cache systems		
On CPU	Good	High
External	Very good	Very high
On DRAM	Very good	Low

The future trend and needs for DRAMs, in short, call for realization of high-performance memory systems. These systems must meet the demands posed by ever-advancing processor performance, based on the optimization of the whole chip. We must reconsider the usefulness of the present standard DRAMs, especially for main memory uses in low-end computers. DRAMs must keep pace with the evolution of the MPUs and also fully provide the advantages of low-cost, high-density memory cores.

With these technical issues and background in mind, we propose, and have fabricated, a DRAM with an on-chip cache, which we call the Cache DRAM.[3] Our main goal is to explore the possibilities and feasibility of a new approach in application-specific memories.

Bridging the speed gap inexpensively

In Table 1 we compare several possible measures to overcome the speed-gap problem in terms of performance and cost. High-performance MPU systems may be covered by external cache systems with high-speed static RAMs (SRAMs) or main-memory systems with high-speed Bi-CMOS DRAMs. But these schemes lose the advantages of low-cost, high-density DRAM memory cores. Hence they are not adequate for cost-sensitive low-end systems. We need a high-performance but cost-effective memory system for performance-oriented but cost-sensitive application areas. A scheme integrating the cache on a DRAM chip optimizes cost/performance with architectural improvements in DRAM main memory. We discuss some basic considerations for these improvements next.

To enhance the performance of standard DRAMs and reduce the total system cost, we examined a memory-

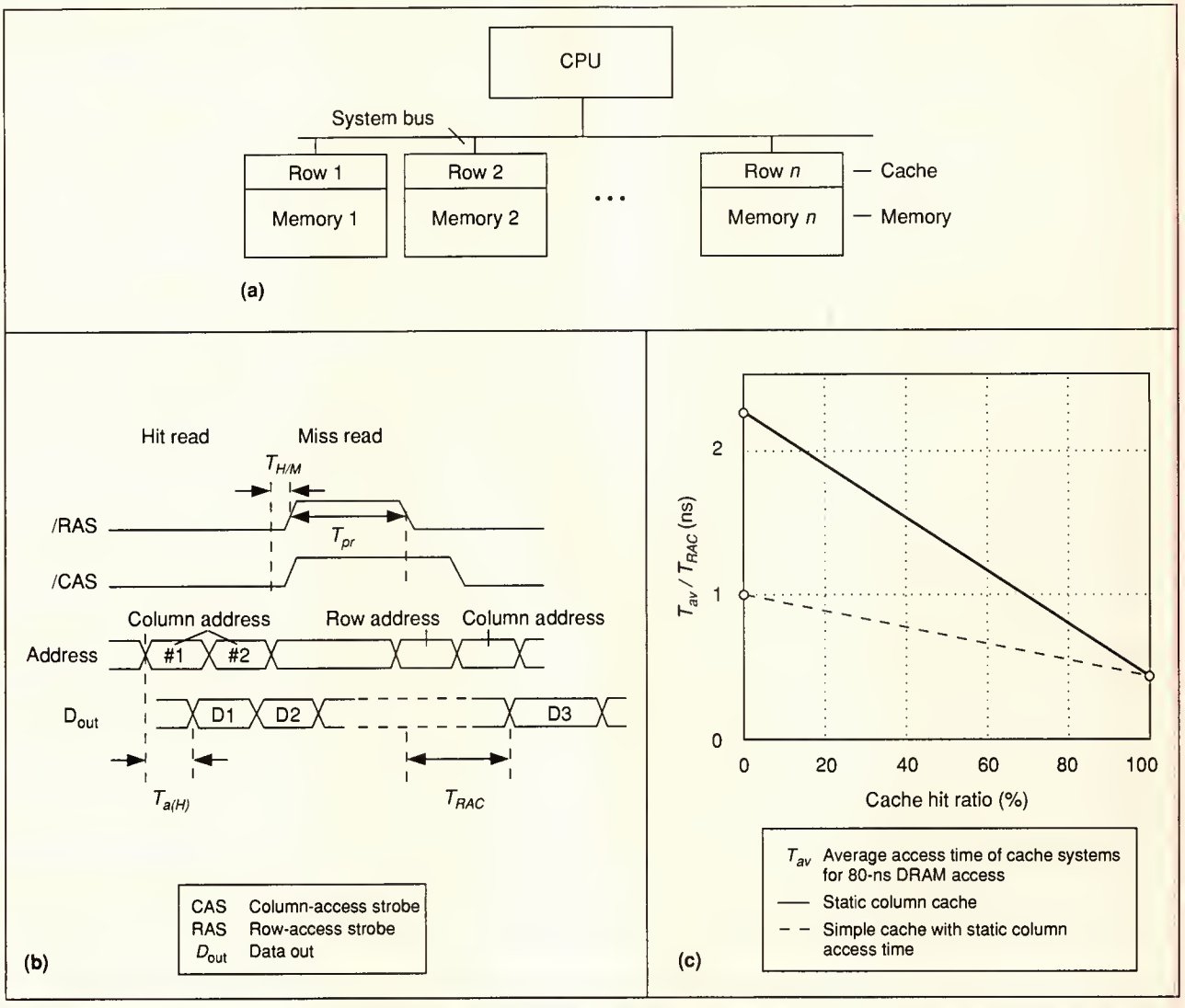


Figure 2. The static column mode organization (a), operation as a cache (b), and its performance improvement as a cache (c).

based cache scheme. This scheme involves a cache associated with the main memory rather than the processor. Originally this approach offered advantages of faster main memory and no consistency problem in multiprocessor systems.[4] Instruction caches have been implemented on MPU chips successfully, but data caches have not. The memory-based cache seems to be a promising candidate for realizing a good combination of the data cache on the main memory and the instruction cache on MPU chips. The shortcomings of the memory-based cache approach are:

1) cache access is slow due to the delay on the system bus, and
2) cache capacity is inevitably large because it is proportional to the memory capacity; so it's an expensive approach.

But in a system with one or only a few DRAM chips, the first concern vanishes. Placing the memory-based cache on the DRAM chip alleviates the second cost problem. So in this type of application, the situation changes drastically with the cache implemented on DRAMs. With a simple cache control, the memory-based cache scheme suggests an optimum form for future main memory designs.

In the design of conventional main memory systems using standard DRAMs, the static column mode or the page mode functions as the built-in cache.[5] The static column mode (or page mode) of DRAMs provides high-speed access to data in a row latched in the sense amplifiers. This is the simplest form of the memory-based cache. Figures 2a and 2b display the operation of this scheme. The static column cache has disadvantages in the hit ratio and miss penalty, as shown later.

Note that in Figure 2b, the DRAM is in the active state at the beginning of the access cycle, and the miss cycle begins with the DRAM precharging operation. The static column approach produces a large miss penalty because of this long DRAM precharging time, thus making the minimum-required hit ratio rather high. To achieve the advantage of the static column cache, the average access time must be less than the usual DRAM access time. Therefore, considering the additional hit/miss judgment time $T_{(H/M)}$, the following relation must hold:

$$r_H T_{a(H)} + (1 - r_H)[T_{pr} + T_{RAC} + T_{(H/M)}] < T_{RAC}$$

where r_H is the hit ratio, $T_{a(H)}$ is the hit access time (static column mode), T_{pr} is the precharging time, and T_{RAC} is the /RAS access time.

Let's assume in Figure 2b that T_{RAC} and T_{pr} each equal 80 ns, $T_{a(H)}$ is 40 ns, and typically $T_{(H/M)}$ is 20 ns. Then, the minimum-required hit ratio becomes 75 percent. Figure 2c shows that performance improves less with a static column cache than with the usual cache. The static column cache enhances performance about 30 percent for a hit ratio of 90 percent, while the performance improvement of the usual cache is 45 percent. The large DRAM precharging time degrades performance of the static column cache.

Smith[4] reports many discussions of the relation between the cache memory size and cache hit ratio for various cache organizations. As this relation strongly depends on the executed programs, many opinions are expressed. Figure 3 shows one example of the data cache hit ratio.[5,6] In this figure, the hit ratio in the static column mode is typically about 70-80 percent. In the static column approach, the cache inevitably becomes one way and direct mapped (each main memory location can be mapped onto only one cache location) with a very large block size (more than 1 Kbit/chip). Therefore, the cache hit ratio is not so high.

This ratio changes significantly according to cache organization features such as the number of ways and the block size. In the static column approach, the block size is too large for the total cache capacity. So, in the case of an access address transfer out of a cache entry, too few cache entries are available to be accessed next, often causing a cache miss. Therefore the static column approach may produce a rather poor cache hit ratio. But the figure also shows that a little improvement in the cache organization, such as a multiple-way organization and reduction in the block size, makes the cache hit ratio much better.

These data show that the minimum required hit ratio for the static column mode is often higher than the one achieved in the actual static column cache. From all these discussions, an improved cache scheme on a DRAM chip suggests the optimum scheme for small systems.

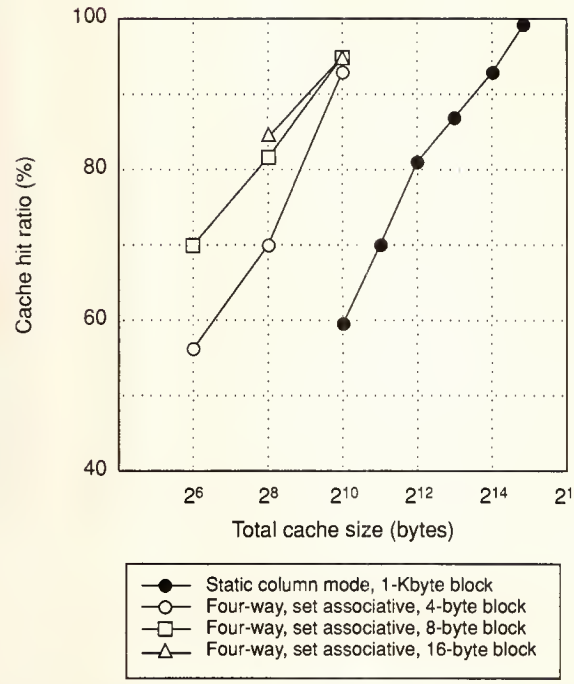


Figure 3. Cache hit ratio vs. cache organization (VAX 11/790 data cache).

The Cache DRAM concept

To implement an improved cache scheme on a DRAM chip effectively, we adhered to certain design considerations. Considering future small and high-performance systems, we see the following advantages of integrating the cache memory on the DRAM chip:

• *Improvement of the average access time.* While the T_{RAC} (/RAS access time) of a DRAM is large, a secondary access mode to the on-chip cache using only column addresses enhances the access speed. Increasing the hit ratio and eliminating the miss penalty improves the average access time significantly.

• *High-speed data transfer between the DRAM and the cache memory.* Limitations of the system bus width and chip I/O rate compel block data to be transferred serially, for example, by using nibble mode. With a wide internal data bus on the DRAM chip, we achieve the block data transfer between the DRAM core and the cache concurrently, in one cycle, without affecting the chip I/O count.

• *Improvement in the miss penalty.* By waiting in the standby state at the beginning of the memory access cycle, we can eliminate the precharging time (T_{pr}) from the access time in a miss. Executing the block transfer in parallel as described earlier enhances the block transfer rate in a miss cycle.

• *Introducing ECC without performance degradation.* Mano et al.[7] and Kotani et al.[8] reported DRAMs

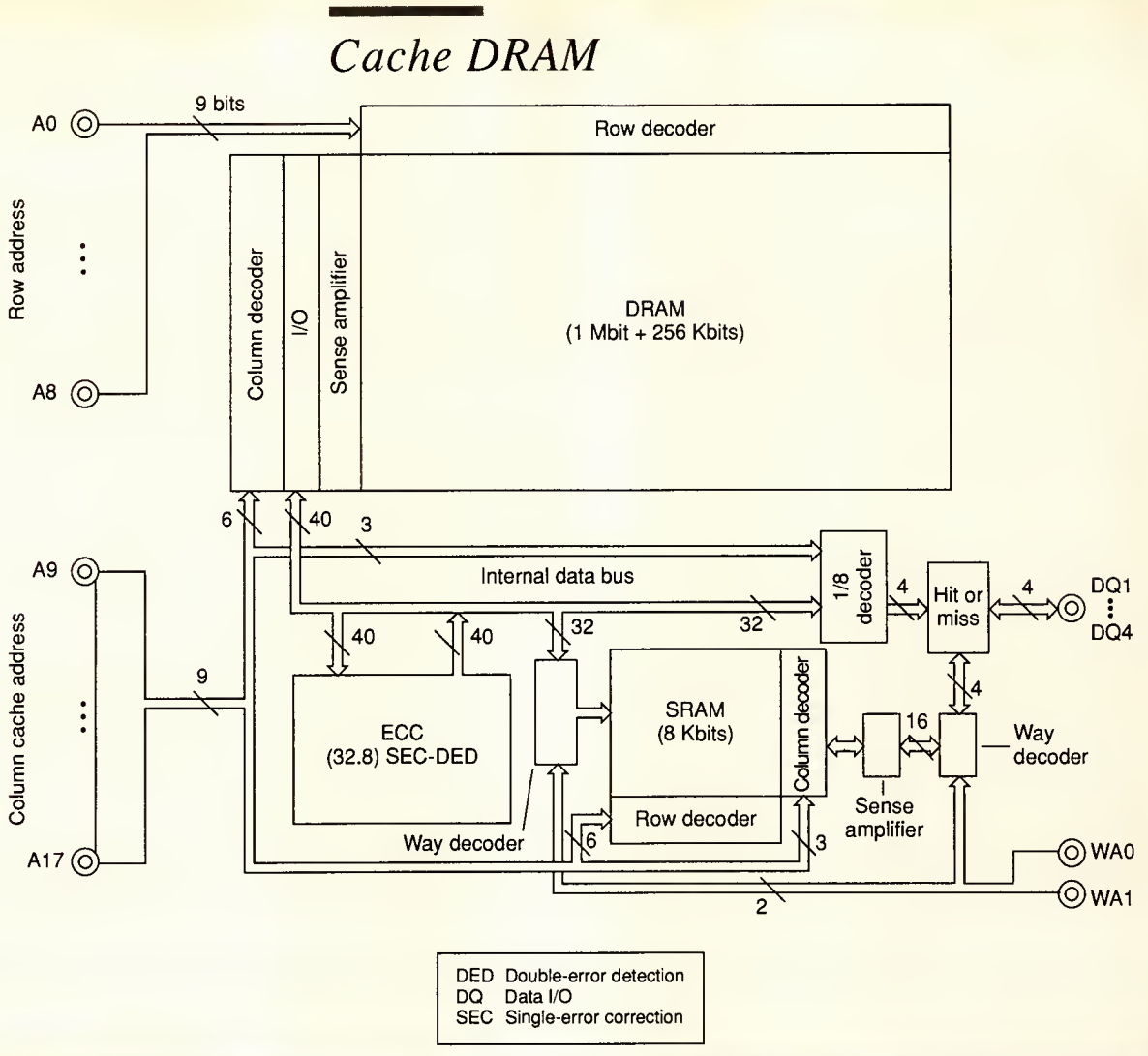


Figure 4. Block diagram of the Cache DRAM.

with on-chip ECC to support DRAM data reliability, especially for soft-error immunity. Though the ECC operation usually delays DRAM access to detect and correct errors, the delay is hidden because access occurs mainly to the cache memory. ECC implementation on the DRAM chip is easy and inexpensive.

The requirements and our performance goals for the Cache DRAM include:

- Achieving a reasonable typical cache hit ratio (>90 percent) with minimum chip area penalty.
- Ensuring cache access comparable with or faster than that possible in the static column mode. (Some special designs of access paths to the cache, different from that of standard SRAMs, should be employed for faster cache access.)
- Reducing the miss penalty.

One of the benefits of the cache system is to lower the busy rate of the system bus, especially in multiprocessor systems. To open the system bus when the cache is accessed, we favor a multiport scheme (separate buses for DRAM and cache). We also reconsidered the current address multiplex scheme found in standard DRAMs. Address multiplexing imposes extra timing restrictions especially on cache decoding. For an effective DRAM decoding scheme, address multiplexing should be abandoned in some cases. Another main consideration centered on the peaceful coexistence of cache and DRAM memory arrays in terms of the fabrication process.

In all, the cost/performance optimization of course remains the major concern. All of these features define our Cache DRAM concept.

A Cache DRAM architecture

Figure 4 shows the block diagram of a Cache DRAM with 1 Mbits (256 Kwords × 4 bits) of main memory. We constructed this RAM to be suitable for a four-way set-associative scheme and a block size of 8 bits for each DQ (I/O pin). As a result, a good cache hit ratio

can be expected. The RAM contains an additional 256 Kbits of DRAM cells for ECC and an 8-Kbit, high-speed SRAM for cache memory.

We avoided the address multiplex scheme to eliminate the extra timing restrictions on Cache DRAM operations. We divided the 20 total bit addresses (A0-17 and WA0-1) into A0-A8 row addresses and A9-A17 column addresses for the DRAM. Of these column addresses, A9-A14 function as set-selection addresses and A15-A17 as bit-decoding addresses in a block. The SRAM cache is fully decoded by the nine A9-A17 cache address bits, which correspond to the column addresses of the DRAM, and the additional two cache way addresses WA0 and WA1. The A9-A14 row addresses and A15-A17 column addresses of the SRAM , correspond to those of the DRAM.

This direct physical correspondence between DRAM and SRAM organizations contributes to the fast access operation, as we describe later. The capacity ratio of SRAM to DRAM is fixed but alterable by a simple extension of the memory mat organization of this design. In a system using the Cache DRAM, the cache capacity increases proportionally to that of the DRAM, which realizes reasonable expansion of the main memory organization.

Figure 5 more specifically shows the data-mapping relationship between the DRAM and SRAM. The DRAM array consists of five planes of 256 Kbits each, four of which are the data bit planes and one of which is the parity bit plane for ECC.

The SRAM array consists of four 2-Kbit planes. Each SRAM plane divides into four blocks corresponding to the cache way organization; each are selected by the way address. The block size is 8 bits per I/O. Thus we organized the cache as 8 bits × 64 sets × 4 ways × 4 I/Os for 8 Kbits. By this simple implementation of cache memory, we expect to achieve a hit ratio of more than 90 percent.

The DRAM data memory mat corresponds to the SRAM cache block/set/way organization, as shown in Figure 5. Eight-bit blocks of data from each DRAM plane, a total of 40 bits, are read out in parallel. They transfer to the SRAM and ECC unit via a 40-bit-wide internal data bus. After checking and correcting any errors, the 32 data bits transfer to the selected location in the SRAM plane, and the 40-bit word transfers to the DRAM plane, as described earlier. The data transfers in one internal DRAM read cycle, enhancing the data rate significantly.

The Cache DRAM has four basic access cycles:

- *Hit read* reads data from the cache memory;
- *Hit write* writes data to the cache memory and the corresponding DRAM bits (write-through cache);
- *Miss read* transfers (maps) read data from the DRAM and the block data, including the accessed bits, to the corresponding cache location; and

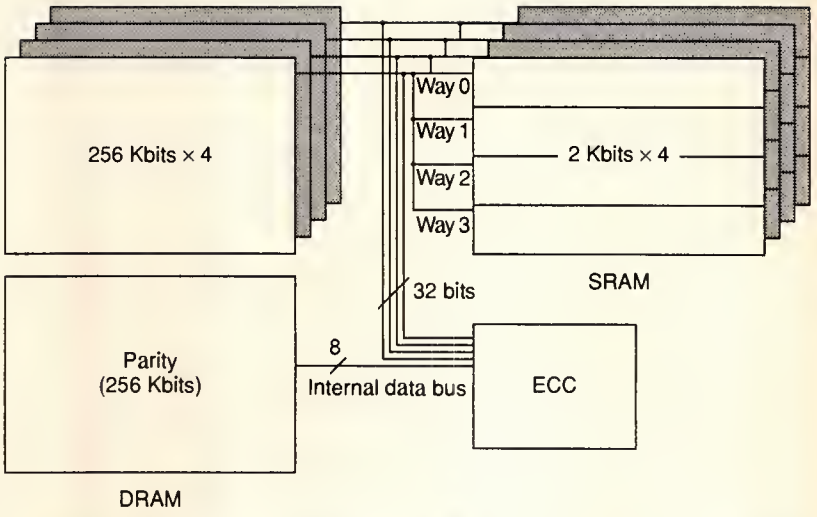


Figure 5. Data-mapping relation between DRAM and SRAM.

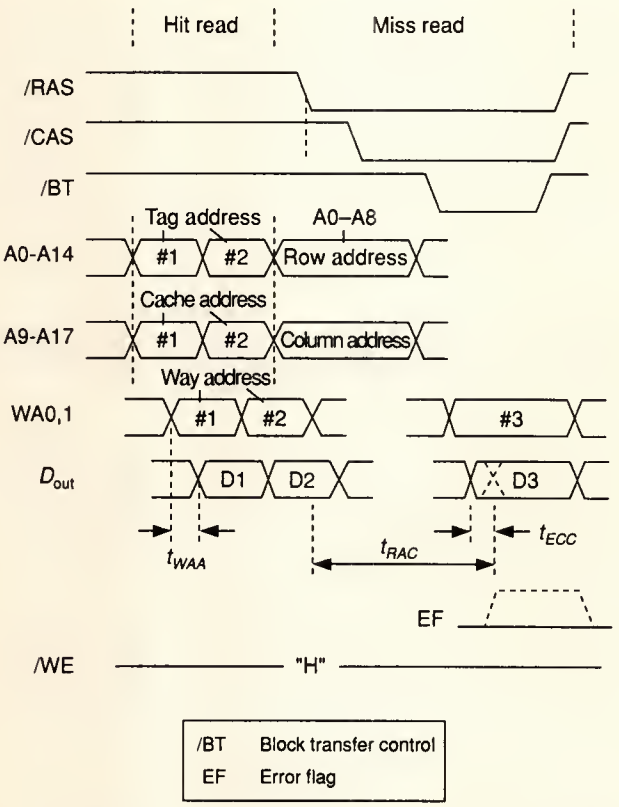


Figure 6. Timing chart of Cache DRAM operation.

- *Miss write* writes data to the DRAM, then either transfers or does not transfer data to the cache memory, as selected by an external signal /BT.

Figure 6 contains a timing chart for basic operations. Besides ordinary DRAM/SRAM control, an external

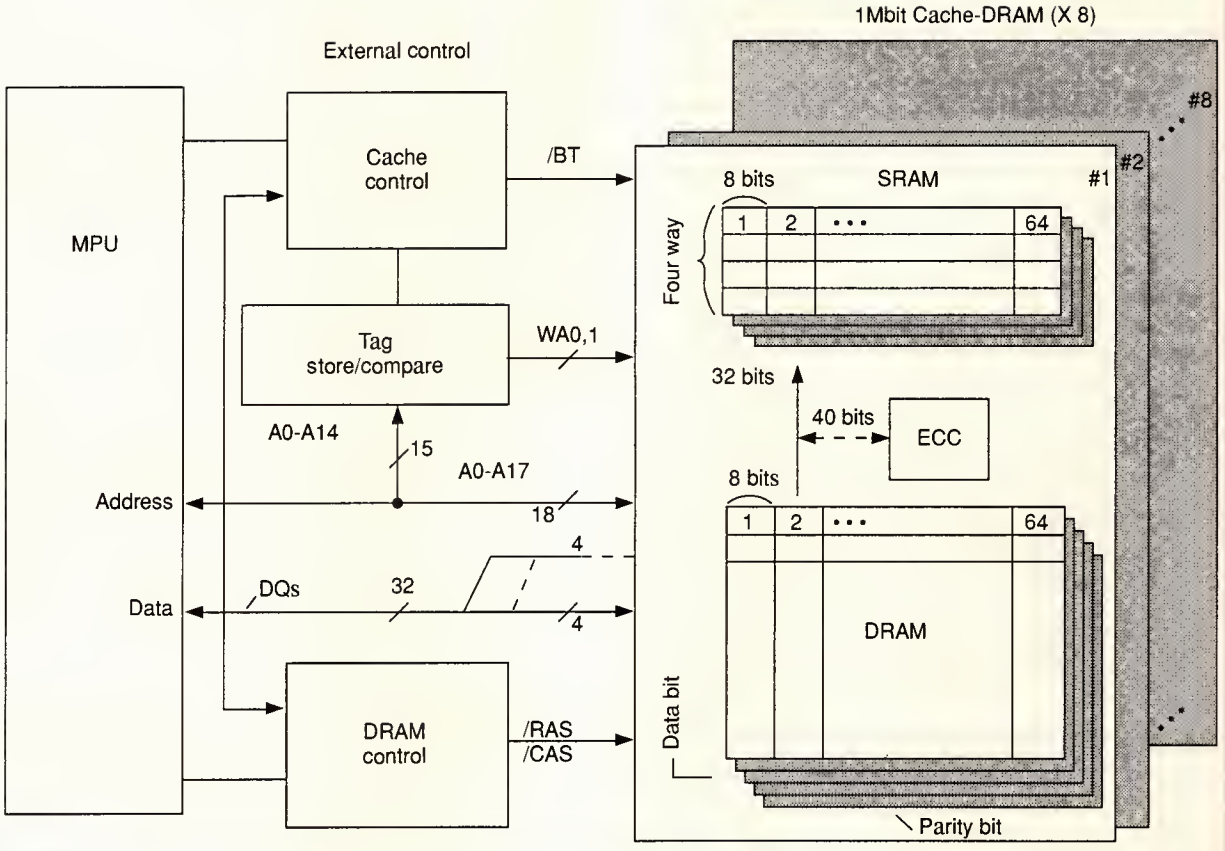


Figure 7. System application.

input signal /BT controls the data transfer. /RAS, /CAS, /WE, and /BT signals control the basic four modes of operation. The access cycle begins with /RAS = *H* (DRAM standby state) to eliminate the miss penalty caused by the DRAM precharging time.

Figure 7 shows one example of a system configuration using the Cache DRAM. The eight Cache DRAM chips in this system compose a 1-Mbyte memory system for a 32-bit-wide system bus. In this case, the cache block of 32 bits × 8, or 256 bits, is about a nominal size in various systems. We can easily alter and select the cache block size by bonding options (8/4/2 bits per DQ) for a wide range of system applications. When accessed, 18 address bits, A0-A17, transfer to the Cache DRAM. At the same time the 15 A0-A14 address bits are compared with the cache entry addresses in the tag memory. In the Cache DRAM, nine address bits (A9-A17) transfer to the SRAM as cache addresses, and 16 bits of data corresponding to the cache address are read out to the final stage of the data output path. This parallel system operation hides most of the hit/miss judgment time.

In the case of a cache hit, the Cache DRAM receives the two WA0 and WA1 cache way addresses from the cache controller. One of the four sets of data waiting at the final stage is selected and output. Thus the WA0 and WA1 way addresses can access the data in the final stage very quickly.

In a cache miss, normal DRAM access starts with the /RAS-/CAS cycle. The 15 A0-A14 address signals select one 40-bit ECC word and send it to the ECC unit on the internal bus lines. After checking and correcting an error, the external signal /BT transfers the 32-bit data to the way in the SRAM plane selected by the two cache way-address bits, WA0 and WA1. The replacement logic in the cache controller generates WA0 and WA1. At the same time, the 1/8 decoder selects four of 32 corrected data bits via the three A15-A17 address bits and outputs them.

We adopted a write-through scheme in the Cache DRAM. In the hit-write case, the controller writes data to the DRAM and cache memory in parallel. However, it is not difficult to change the hit-write scheme to copy back. Also we easily bypass the cache by ensuring that /BT = *H* in the DRAM access cycles.

The controller needed for this Cache DRAM is simple. Most of the commonly available cache controllers fit this configuration, and the controller circuitry

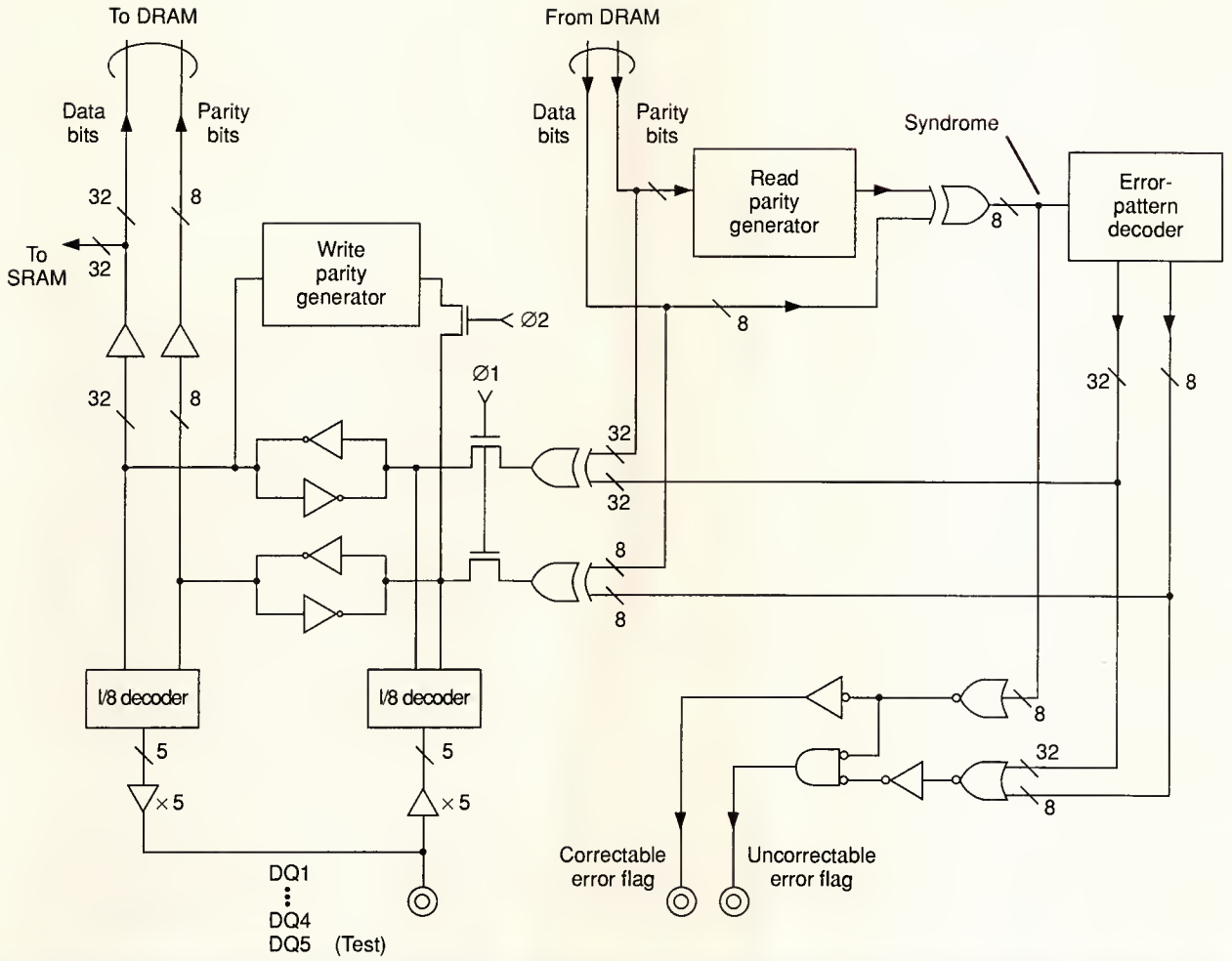


Figure 8. The ECC circuit.

may be implemented on the Cache DRAM chip itself.

ECC scheme

With the wide internal data bus for cache data transfer described earlier, we can easily implement on-chip ECC circuitry in the Cache DRAM. The ECC implementation improves DRAM data reliability, especially soft-error immunity, with a small penalty for area and performance. The ECC function works in any DRAM cycle, that is, read, write, and /CAS-before-/RAS refresh cycles. One ECC word consists of 32 data bits and 8 parity bits. We adopted the modified Hamming code (odd-parity code) to correct a single error or detect double errors in each ECC word. The ECC unit outputs corrected data to DQ pins and writes to SRAM memory cells; the unit also rewrites the data to the DRAM memory cells in any of the DRAM access cycles.

The ECC circuit shown in Figure 8 conducts a read-modify-write operation internally in a read cycle and a /CAS-before-/RAS refresh cycle. The circuit checks and corrects errors in one 40-bit ECC word, read out onto the internal bus lines, through the read parity generator, Exclusive-Or circuits, and the error-pattern decoder. The read parity generator and the first Exclusive-Or circuits generate the error syndromes.

The error-pattern decoder decodes the syndromes, and the second Exclusive-Or circuits correct the error. The corrected 40-bit word is then written back into the DRAM cells, and at the same time the data transfers to the output pins and/or the SRAM.

In the write cycle, the ECC unit also conducts an internal read-modify-write operation. At first, 32 data bits and 8 parity bits are read internally, and any errors are corrected, just as in the read cycle. Then 4 new data bits from four DQs are written into the data latches, and 8 new parity bits are regenerated from the new 32

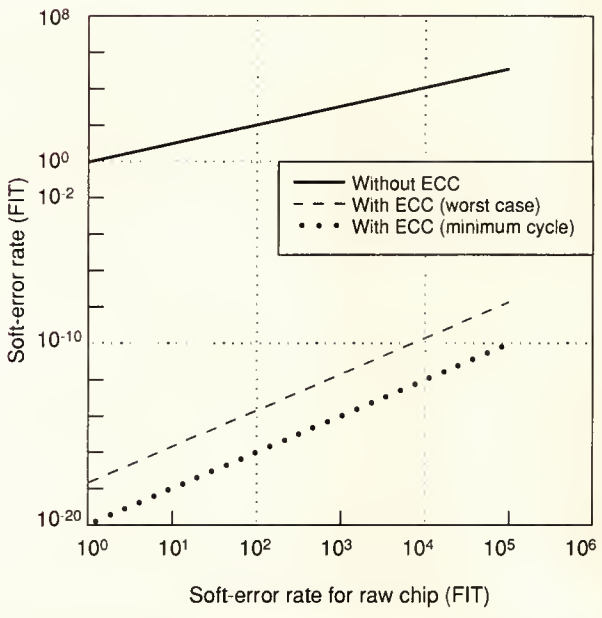


Figure 9. Soft-error rate with and without ECC.

data bits in the write parity generator. Finally, the ECC unit writes this new word of 40 bits into the DRAM cells.

To test the ECC circuit, the ECC functions can be disabled by an ECC control signal (ECD). Writing ECC words including errors with the ECC disabled and then reading data with the ECC enabled tests the ECC circuit function. Moreover, in the test mode, the parity bit can be accessed directly, bypassing the ECC circuitry, and we can test the DRAM as a 256-Kword × 5-bit RAM. Furthermore, to initialize the DRAM data, the test mode circuitry activates a reset mode with a flash write operation. This step eliminates any ECC code contradiction before operating the RAM. In operation, if any error occurs, an error flag (EF) and/or an uncorrectable error flag (UCE) are set.

Figure 9 depicts an estimation of the soft-error rate of the typical 1-Mbit DRAM with and without ECC. The figure assumes a 1-Mbit DRAM organization and the ECC scheme just described. The figure's dashed line corresponds to a worst case in which an error is corrected only during the refresh cycle with a maximum refresh interval (15.6 μs). The dotted line corresponds to a best case in which the ECC checks an error every cycle at a minimum interval (160 ns). The actual ECC soft-error rate appears in the region between the broken and dotted lines. The improvement in the soft-error rate depends on the error rate of the raw chip. Typically, the soft-error rate of a raw chip is less than 1,000 FIT (Failure in Term). As this figure shows, we can expect more than a 10-order-of-magnitude improvement in the soft-error rate with this ECC scheme.

Experimental results

Based on these concept and design considerations, we fabricated an experimental Cache DRAM. A photomicrograph of the RAM appears in Figure 10, and the box summarizes the chip characteristics. This chip measures 6.61 × 17.35 square millimeters.

The fabrication process is the same as the conventional DRAM process: a P-type substrate, 1.2-μm, twin-

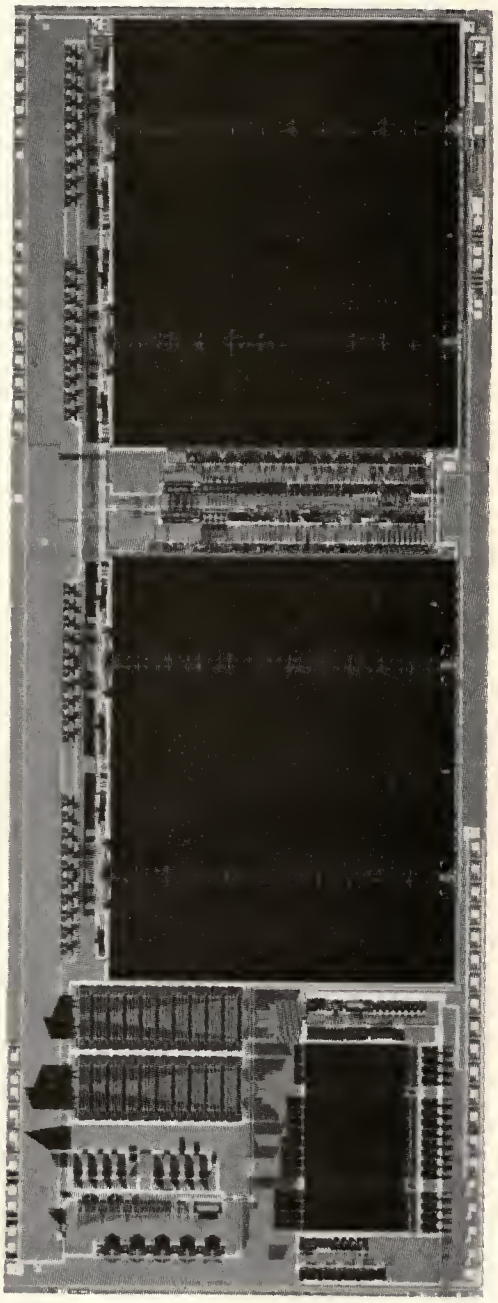


Figure 10. Chip photomicrograph.

well CMOS with triple-polysilicon and single-metal (aluminum) layers. The normal DRAM process is not compatible with that commonly used for SRAM memory cells of highly resistive polysilicon loads. But a normal DRAM process realizes full-CMOS SRAM cells composed of a PMOS transistor load. We used the relatively large full-CMOS SRAM cell because of the small extra chip area for SRAM cells. The realization of this compatible process eliminates extra process steps, which significantly reduced the process cost.

The DRAM cell measures 3.4 × 8.8 sq μm, and the SRAM cell fits into 14 × 21 sq μm. These cell sizes are typical for standard 1-Mbit DRAMs and 256-Kbit SRAMs. The Cache DRAM requires 37 signal and power supply pins and can be housed in a 400-mil, 40-pin SOJ (small-outline, J-leaded) package.

In a cache hit cycle, we observed an address access time of 12 ns and a way-address access time of 7 ns. In a cache miss cycle, including the ECC access penalty of 12 ns, we obtained an 80-ns access time for /RAS access. As already described, the parallel system operation hides the hit/miss judgment time. Therefore we estimate the average access time to be less than 20 ns under the condition that the hit ratio is 90 percent. The block data transfer of 8 bits per DQ completes in one 160-ns miss cycle. The Cache DRAM transfers block data about 3.5 times faster than the standard DRAM design with nibble mode data transfer.

We adopted a macrocell design method for this chip that is composed of the DRAM, SRAM, and ECC blocks. The DRAM core follows standard 1-Mbit DRAM design, but we designed the SRAM, ECC circuitry, and interface circuitry in a modular manner. While we have not optimized the chip layout for the chip size, we estimate it can be reduced by 30 percent in an optimized, more square layout.

It took us 10 man-months for architecture definition and chip design, a very short time due to the design method. Even though we used a handwritten layout for the SRAM memory array and interface design, this cellular design produced prototypes efficiently because the DRAM core—including the peripheral control circuit—was previously optimized for standard DRAMs.

In combination with more advanced design methods such as a RAM generator and placement/routing tools for SRAMs and peripheral circuitry, we expect to experience further design improvement. We found the DRAM core however to be very difficult to implement with automatic generation tools. The inflexible DRAM memory array does not lend itself to reconfigurations of the word/bit organization, and the complicated control circuit should be fully customized.

Chip Characteristics

Cache (SRAM)

Capacity	32 bits × 64 sets × four ways (= 8 Kbits)
DQ	× 4
Organization	Four-way, set-associative, write-through
Block size	8 bits × four DQs (= 32 bits)
Address access time	12 ns
Way addresss access time	7 ns
Operating current	31 mA (*T* cycle = 60 ns)
Cell size	14 × 21 sq μm (full CMOS cell)

DRAM

Capacity	1 Mbit (data) + 256 Kbits (parity)
DQ	× 4
Access time	T_{RAC} = 80 ns (including T_{ECC} = 12 ns)
ECC code	(32, 8) Single-error correction/ double-error detection
Operating current	42 mA (*T* cycle = 160 ns)
Cell size	3.4 × 8.8 sq μm
Process technologies	1.2-μm, twin-well CMOS, triple-polysilicon/single aluminum
Chip size	6.61 × 17.35 sq mm

We have proposed the Cache DRAM concept of integrating cache memory on a DRAM chip, and we have shown the results obtained from a fabricated experimental device. This experimental device let us develop the core technologies required to integrate cache memory with a standard DRAM. We limited the DRAM capacity to 1 Mbits and used a previously established process to design the Cache DRAM. In so doing, we confirmed our design choices of 1) a high-speed SRAM using a DRAM process, 2) a high-speed block data transfer, and 3) soft-error reduction with an on-chip ECC.

In the near future we plan to implement an entire main memory in one chip for small systems such as low-end workstations and personal computers. High-speed hierarchical memory organization is a key factor for this purpose. In the era beyond the 16-Mbit DRAM, a single-chip main memory system with integrated tag memory, cache replacement logic, and a DRAM controller will play an important role. This is a natural evolutionary form of the Cache DRAM.

Table 2.
Application spectrum of the Cache DRAM.

System	Application
Low end; no wait-state memory access	High-performance MPU systems RISC systems Personal computers with one-chip main memory
High end; secondary cache scheme	Superminis Mainframes
Special purpose	Video LSI testers

The realization of a hierarchical RAM chip containing a whole main memory is our dream, but this dream is about to come true. The Cache DRAM concept also works effectively in high-end computer systems with secondary cache schemes and in special-purpose systems such as videos, LSI testers, and so on. Table 2 lists some suggested application areas of the Cache DRAM.

In the present VLSI/ULSI age, the benefits of integration itself must be explored in every aspect of the evolving computer systems, including memory systems. One lesson we've learned from this project is that the hierarchical approach smashes the cost, which has been the central dogma of present computer systems.

Acknowledgments

We wish to thank H. Komiya, T. Nakano, S. Kayano, T. Yoshihara, and S. Satoh for their encouragement. We are also indebted to Y. Tanaka, K. Tanaka, and Y. Nakaoka for the useful discussions we had with them.

References

1. W.M. Regitz et al., "A Three-Transistor-Cell, 1024 bit 500 ns MOS RAM," *ISSCC Digest of Technical Papers*, Feb. 1970, pp. 42-43.
2. T. Watanabe et al., "An 8K Byte Intelligent Cache Memory," *ISSCC Digest of Technical Papers*, Feb. 1987, pp. 266-267.
3. M. Asakura et al., "An Experimental 1Mb Cache DRAM with ECC," *Symp. VLSI Circuits Digest of Technical Papers*, 1989, pp. 43-44.
4. A.J. Smith, "Cache Memories," *Computing Surveys*, Vol. 14, No. 3, 1982, pp. 473-530.
5. J.R. Goodman et al., "The Use of Static Column RAM as a Memory Hierarchy," *11th Int'l Symp. Computer Architecture*, 1984, pp. 167-174.
6. M.D. Hill et al., "Experimental Evaluation of On-Chip Microprocessor Cache Memories," *11th Int'l Symp. Computer Architecture*, 1984, pp. 158-166.
7. T. Mano et al., "Circuit Technologies for 16Mb DRAMs," *ISSCC Digest of Technical Papers*, Feb. 1970, pp. 42-43.
8. H. Kotani et al., "4Mbit DRAM Design Including 16-Bit-Concurrent ECC," *Symp. VLSI Circuits Digest of Technical Papers*, 1987, pp. 87-88.

Hideto Hidaka is a senior engineer at Mitsubishi Electric Corporation's LSI R&D Laboratory, where he has been engaged in the research and development of high-density DRAMs and the Cache DRAM. Previously he was a research affiliate at the Massachusetts Institute of Technology's Media Laboratory.

Hidaka received his BS and MS degrees in electronic engineering from the University of Tokyo and is a member of the IEEE Computer Society.

Yoshio Matsuda, assistant manager of the Laboratory, develops MOS VLSI DRAM circuit design. Currently, he works on developing the Cache DRAM and a 16-Mbit DRAM. Matsuda received his BS degree in physics and MS and PhD degrees in applied physics from Osaka University.

Mikio Asakura, a Laboratory researcher, has been engaged in the development of the Cache DRAM and MOS VLSI DRAM circuit design. Asakura received his BS degree in physics and an MS degree in nuclear engineering from Kyoto University, Japan. He is a member of the Institute of Electronics, Information, and Communication Engineers of Japan.

Kazuyasu Fujishima, manager of the Laboratory's ULSI dynamic RAM Design group, has been engaged in the development of 64-Kbit to 16-Mbit DRAMs. His technical interests include high-density memory and application-specific memory. He currently works to develop the Cache DRAM and a 16-Mbit DRAM.

Fujishima received his BS, MS, and PhD degrees in electrical engineering from Osaka University.

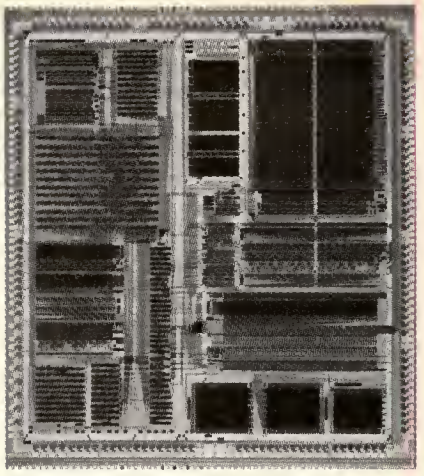
Questions concerning this article can be addressed to Hideto Hidaka, LSI R&D Laboratory, Mitsubishi Electric Corporation, 4-1 Mizuhara, Itami 664 Japan.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153 **Medium** 154 **High** 155

FEATURE



Processing Element Design for a Parallel Computer

To achieve high levels of performance, parallel computers need powerful, compact processing elements. Using VLSI technology in PE design helped us to reach this goal. Here, we analyze overall performance improvement in terms of processor performance and overhead reduction. We also detail the processor architecture and implementation, and solve three essential performance issues for parallel computers.

Katsuyuki Kaneko
Masaitsu Nakajima
Yasuhiro Nakakura
Junji Nishikawa
Ichiro Okabayashi
Hiroshi Kadota

Matsushita Electric Industrial Company, Ltd.

In recent years, progress in VLSI (very large scale integration) technology has increased processor speeds and provided a number of other benefits. This new technology has especially improved the performance and cost-effectiveness of microprocessors and memories in the area of parallel processing. Researchers have concentrated on developing parallel processing for the field of graphics because

- the characteristics of graphics processing—weak correlations among data and locality of data reference—are naturally suitable to parallel processing;
- data operations are rather simple; and
- a high-performance, relatively inexpensive processing element (PE) can operate under these conditions.

For these reasons, designers have realized and tested unique parallel systems such as mesh-type or bus-connected machines.[1-3]

Another promising application of parallel computing is the scientific/technical field, which is mainly represented by fluid dynamic computation, structural analysis, and circuit and device analysis. Especially when the solution of large-sized PDE (partial differential equation) problems is important, researchers have considered many dedicated parallel machines like the Illiac IV.[4-6] However, in contrast to graphics processing, scientific processing includes the following characteristics.

- Systems must provide macrocontrol for convergence analyses or updating time steps.
- Locality of the processing range—which means parallelism can occur—does not always coincide with the convergence speed. Highly parallel schemes (or simple, explicit schemes) are time consuming, while the schemes with fast convergence (or implicit schemes) have less parallelism.
- No PE exists that provides enough memory and performance to solve complex PDE problems in a reasonable amount of time.

For these reasons, it has been difficult to produce good parallel computers in the scientific/technical field. At present, vector pipeline processors dominate. Such machines directly derive advantages from the improved device speeds offered by ECL (emitter-coupled logic) technology. However, rapid improvement is occurring in device characteristics and the integration engineering of CMOS (complementary metal-oxide semiconductor) VLSI technology. Powerful, compact PEs and their networks can reside on just a few VLSI chips. As a result, parallel machines are gaining ground over vector machines.

0272-1732/90/0400-0026$01.00 © 1990 IEEE

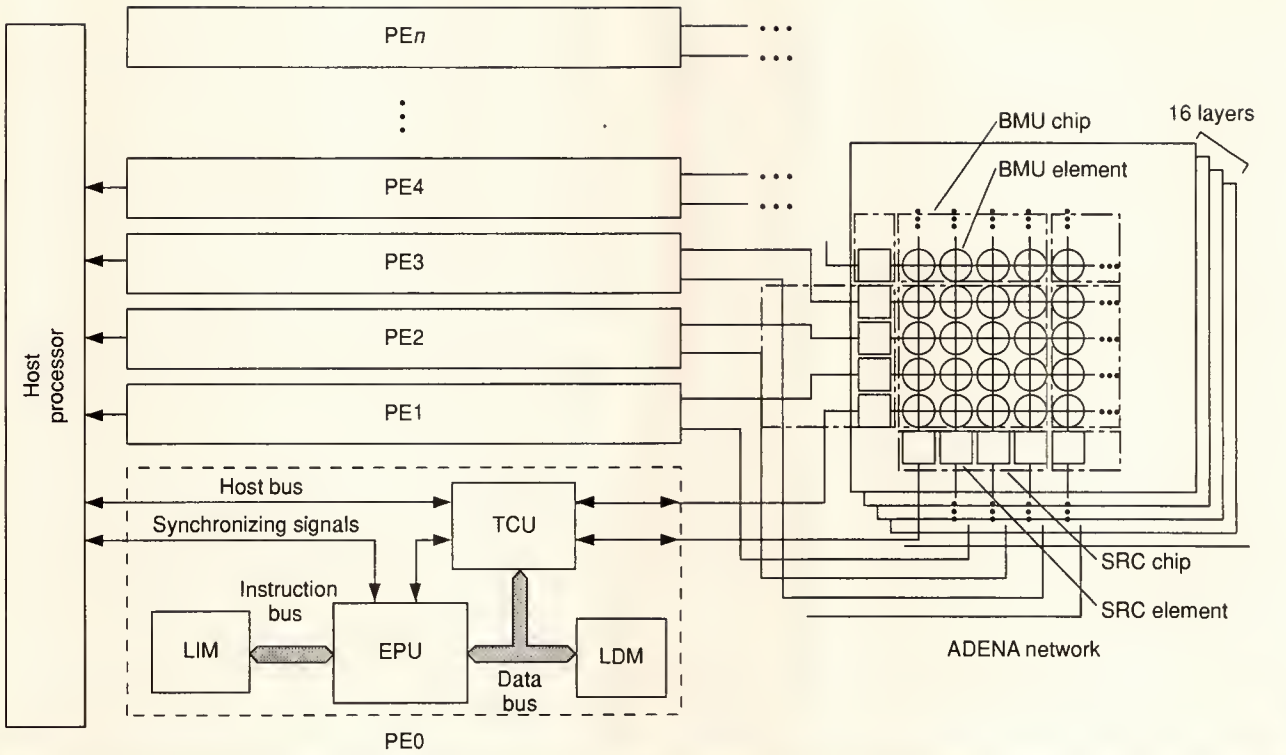


Figure 1. ADENA organization.

Another factor in this development is that performance improvement in vector processors relies mainly on higher operation speeds and clock rates. To achieve these levels of performance, designers must develop technical innovations such as high-speed software and boards, transmission analysis, and cooling technology—in addition to improving ECL technology itself.

Another serious problem is the difficulty of designing large, high-speed memory systems to balance the increasing processing speed of vector computers. As a whole, further development of vector processors requires rather expensive technologies. We do not expect drastic progress in either cost or performance in this field. On the other hand, in parallel systems the operating speed of conventional CMOS microprocessors synchronizes well with that of ordinary memory and peripheral devices. In addition, cost-effective materials and technologies can be used to mount VLSI devices and boards.[7]

The pipeline-pitch ratio between vector processors and current CMOS microprocessors does not exceed 0.01 (several nanoseconds versus a few tenths of a nanosecond). Consequently, in a parallel computer system with a few hundred PEs, the logical peak performance of the system easily exceeds that of vector processors. This is also true for the access time of memory systems, which can reach higher logical, peak I/O rates than those of vector machines.

We have accordingly studied how cost-effectiveness due to the improvement of VLSI technology can apply to a scientific computer system without performance loss. The result is a unique parallel computer, ADENA (Alternating Direction Edition Nexus Array).[8] The core of this array consists of four kinds of VLSI chips, two for PEs and two for the interprocessor network (plus some memory chips).

Here we report the design considerations for the PEs incorporated into ADENA. We especially analyze the factors that limit performance in a parallel processing environment and describe the measures employed to improve these factors at the LSI (large-scale integration) design level.

ADENA overview

Figure 1 presents the basic organization of the parallel computer. The total system consists of 256 PEs, an interprocessor network in which any PE can communicate with any other of the PEs, and the host processor.

The network contains 256 buffer memory unit (BMU) chips and 128 send/receive control (SRC) chips. Each BMU chip consists of a 4×4 array of elements, while the SRC chip holds four elements. We arranged them in 16 layers of 16×16, orthogonally placed BMU elements and 16 network interfaces (SRC elements) on two sides of the array.

The host processor executes system control, data distribution and collection to and from PEs, and other data processing. It also provides users with a programming and debugging environment.

Each PE consists of

- a microprocessor EPU (element processing unit for PEs),[9]
- a network controller TCU (transfer control unit),[10]
- an LIM (local instruction memory) containing six static RAMs that are used as instruction memory for the EPU, and
- an LDM (local data memory) containing 18 dynamic RAMs.

We mounted the EPU and TCU onto a multichip package and packed the memories into modules. Sixteen PEs—each of which consists of the multichip package and three memory modules—and a part of network occupy one board. The core of the parallel computer system consists of 16 boards and their backplanes.

The photomicrographs in Figure 2 show the multichip package and the 16-PE board. The PEs occupy the left side of Figure 2b, while eight SRCs and 16 BMUs occupy the right side. One PE consumes only 42.6 sq cm of the board. The board size is 49.3×35.6 sq cm. The size of the cabinet that houses 16 boards is approximately $46 \times 88.5 \times 91$ cubic cm, which are almost the same dimensions as a desk-side workstation.

The EPU is a RISC (reduced instruction-set computer) with a Harvard bus organization. Employing separate instruction and data buses avoids performance losses caused by von Neumann bottlenecks. The TCU is a multichannel DMA controller that communicates with other PEs via the network and host processor and operates as a coprocessor tightly coupled with the EPU. The host processor stores data into and receives data from the LIMs and LDMs through the TCU. It also controls the EPU by accessing its control registers.

The EPU contains the bus switch that connects the LDM bus to the LIM bus under TCU control. Through the TCU the host computer can deal with all PE resources—the LIM, LDM, and control registers of the EPU and TCU. During normal computation, the TCU acts as a DMA (direct memory access) and network controller under the control of the EPU. The EPU sets the parameters related to interprocessor communication and instructs the TCU to send and receive data to and from other PEs via the network.

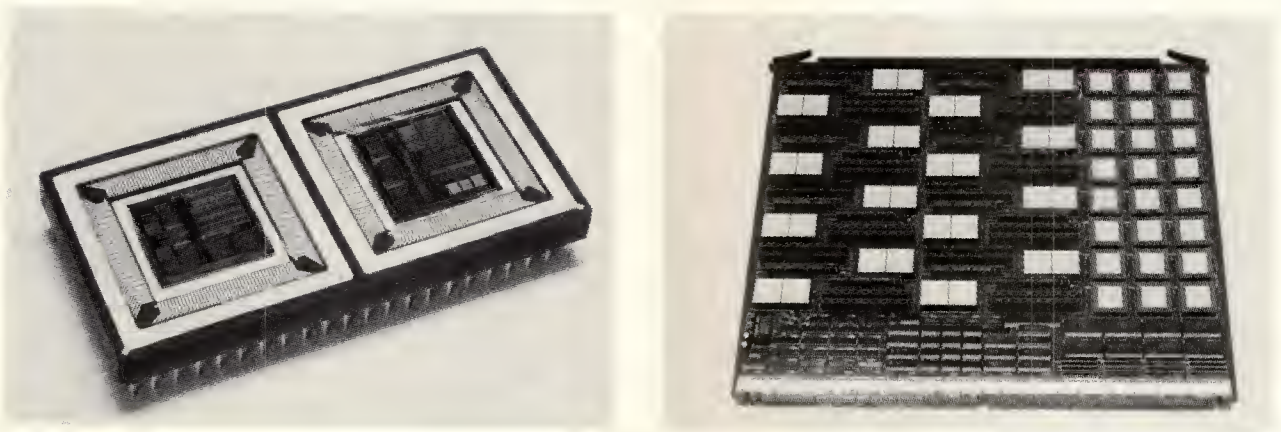
Figure 3 demonstrates the network structure, which reflects the application of VLSI technology.[11] The network consists of only two types of VLSI units, the BMU and SRC. SRC elements act as ports of the network. The network structure contains 16 layers of two-dimensional FIFO arrays connected to a two-dimensional PE array in both straight and twisted forms (see coordinate F's in Figure 3).

Assume that $0 < i, j, k, x, y < 15$. Each PE and R (row) port is identified by its two-dimensional position on the y-z plane of the coordinates shown in the figure. Each C (column) port is identified by its two-dimensional position on the x-z plane. Each BMU element is identified by its three-dimensional position of the x-y-z coordinates. The two-network port of PE (j, k) on the y-z plane connects to the port (j, k) in the y-z plane of the R port and to port (k, j) on the x-z plane of the C port of the network. The BMU and SRC correspond to a 4×4 FIFO array and four network ports, respectively. One layer of the network consists of 16 BMUs and 8 SRCs.

Each board shown in Figure 2—and just described—contains one layer of the network. We designed this network so that when all PEs have one three-dimensional datum, each PE can exchange or "edit" one-dimensional data of one direction from one-dimensional data of another direction. This action can occur while all PEs distribute and gather data from the network in sequential order.

The network allows data transmission between any PEs through two data-editing operations: PE (j, k) > R port (j, k) > BMU (x, j, k) > C port (x, k) > PE (k, x), and PE (k, x) > R port (k, x) > BMU (y, k, x) > C port (y, x) > PE (x, y). Assuming that the data contains an identifier for the destination PE, when PE (j, k) sends data to PE (x, y), PE (j, k) first transmits the data to PE (k, x). At that point, PE (k, x) reads the final destination (x, y) from the data and sends it to PE (x, y).

The network structure in Figure 3 is especially suitable for solving three-dimensional PDE problems using either the ADI (alternating direction implicit) or FFT (fast Fourier transform) methods.[12] ADENA solves PDEs using the ADI method similarly to the way a mesh parallel computer solves PDEs using the Jacobi or the point-Sor (successive overrelaxation) methods.[13] In a mesh computer, each PE is responsible for one grid and contains data corresponding to that grid point. All PEs exchange updated data between their neighboring PEs in every computational step. In ADENA, each PE has responsibility for grid lines of x, y, and z directions and contains data on these grid lines. In each computational step, the PE updates data corresponding to one of three directions—for example x-directional data—and exchanges these data between PEs. Each PE receives these data as other-directional data—for example y-directional data—and computes new y-directional data. All PEs repeat these operations by changing directions, as in $x > y > z > x > y \ldots$.



(a) (b)

Figure 2. The TCU in multichip packaging (a) and a 16-PE-board EPU (b).

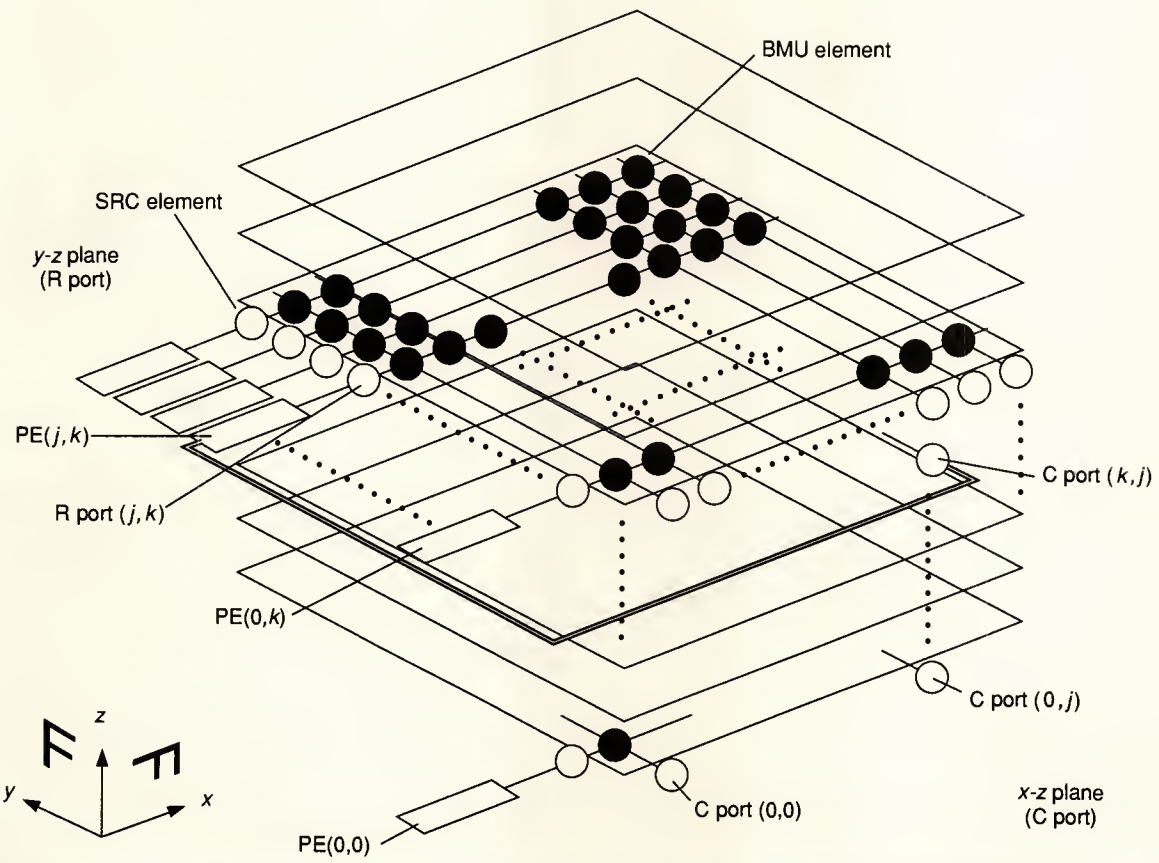


Figure 3. ADENA network structure.

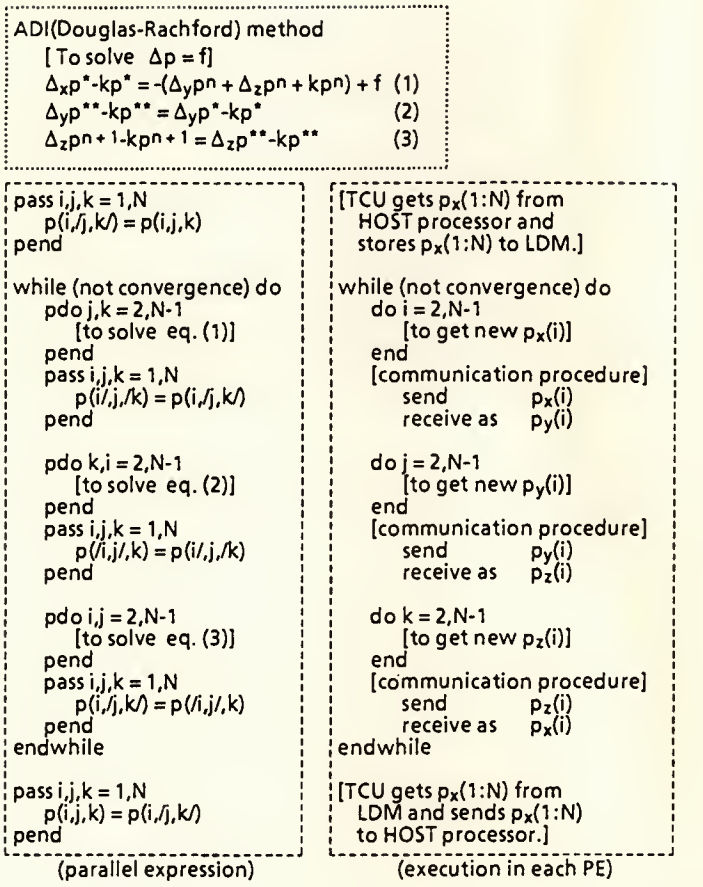


Figure 4. PDE solution in ADENA.

Figure 4 shows the PDE-solution process using the Douglas-Rachford method (a derivative ADI method) in two ways: the expression in a parallel programming language called Adetran (ADENA Fortran) and the execution image in each PE.

Performance analysis

Two methods effectively enhance performance in a parallel computer. The first is straightforward and positive. The second is indirect but includes the elements that are essential to analyze the performance of a parallel processor.

The direct way is to improve the performance of each PE. Both the peak and average performance rates of the system increase naturally in proportion to the corresponding performance of each PE. We designed the EPU to improve both peak and average processing capability.

The processor contains a powerful, double-precision floating-point unit with a multiplier, an add/subtract unit, and 32 words of registers. The EPU also provides an instruction cache, a dedicated control unit for loop operations, an on-chip stack, and other hardware to improve the execution of typical scientific codes. A 64-bit data bus can obtain a high data bandwidth. We discuss and illustrate the organization and architecture of the EPU in the next section.

The indirect way to improve the performance of a parallel computer is to reduce the overhead in several areas that specifically apply to parallel processing. In hypercube machines, the average operation ratio (ratio of computing and communicating time to the total operation period) of each PE is as low as 50 percent, depending on the application and its implementation.[14] The net ratio of true processor computing time to the total operation period is even smaller. Consequently, the amount of overhead caused by the structural factors in parallel processing seriously influences total performance.

Overhead in parallel computers mainly results from task switching, communication, and synchronization.

In the ADENA, the EPU and TCU contain a number of hardware and operational mechanisms to reduce these overhead costs, which we discuss after the next section.

The EPU

We fabricated this RISC with a 64-bit, external data bus and a separate, 24-bit instruction bus using two aluminum-layer, 1.2-micrometer, N-well, CMOS technology.[9] (Reference nine describes the original EPU chip.)

Figure 5 summarizes the internal block diagram. We discuss the most salient features. The chip provides a 64-bit, floating-point data bus with a 4-Mword address space; a 24-bit instruction bus with a 1-Mword address space; a 32-word × 32-bit internal stack memory; and a 256-entry instruction cache with direct mapping control. The floating-point data path includes a 32-word × 64-bit register unit, a double-precision multiplier (Fmul) pipelined into two stages, and a double-precision arithmetic unit (FAU) with the same pipeline structure.

The processor employs a five-stage pipeline scheme

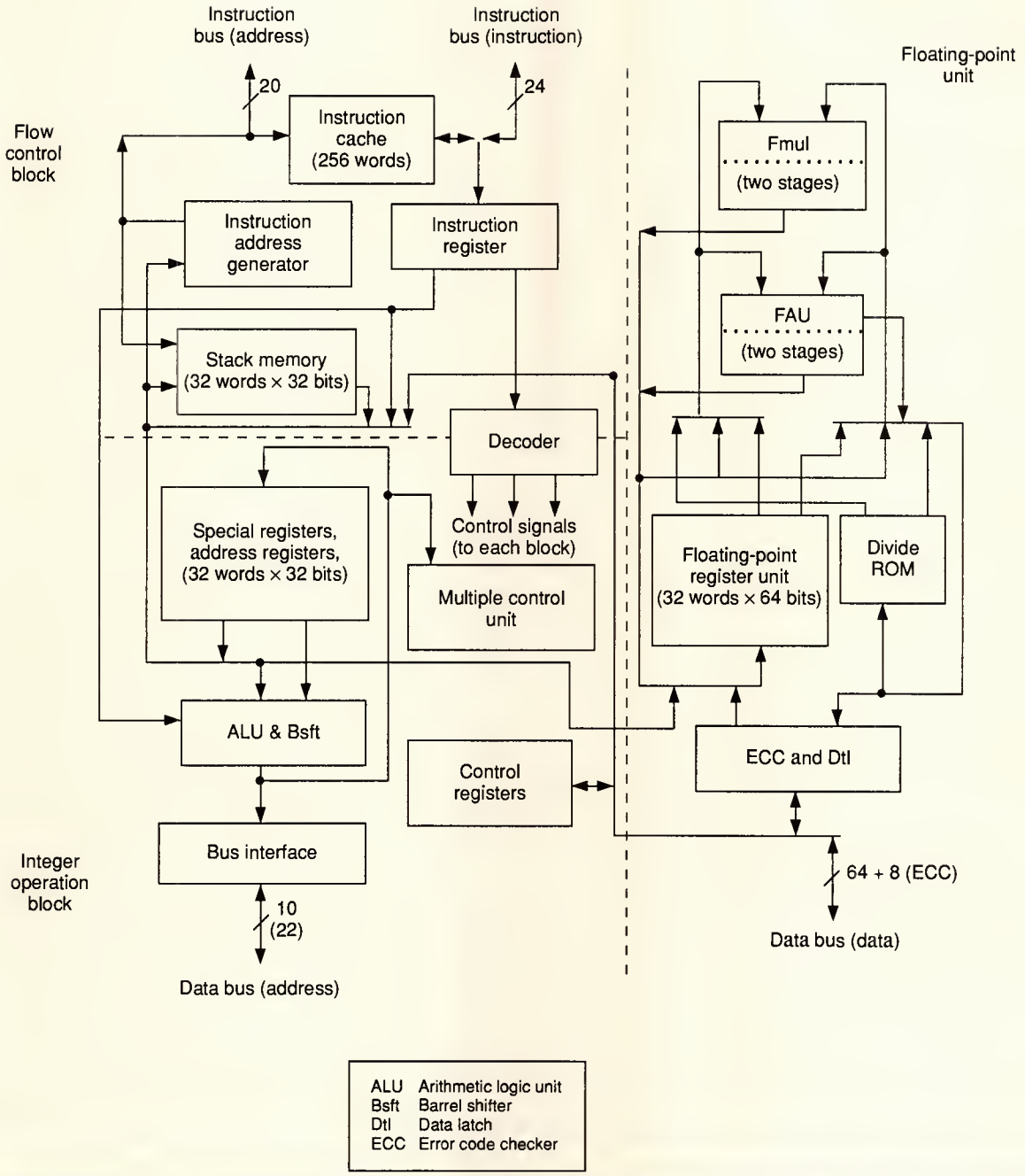


Figure 5. Block diagram of the EPU.

(not shown). Almost all instructions execute in one pipeline cycle. A memory access takes two cycles, and a divide operation using the Fmul, FAU, and divide ROM consumes eight cycles. However, unless a hazardous condition occurs, these instructions do not disturb the instruction flow in the pipeline and execute in one cycle.

Two types of register-manipulation instructions exist: a register-to-register operation with three operand forms, and an immediate data-to-register operation with two operand forms. Instruction fetches and memory accesses execute concurrently because the chip employs isolated instruction and data buses. The instruction and data buses connect to the static and dynamic RAMs without any extra chips. The bus interface block includes page-mode access and refresh control circuits. The simple external interface allows the design of a compact PE for massive parallel computer

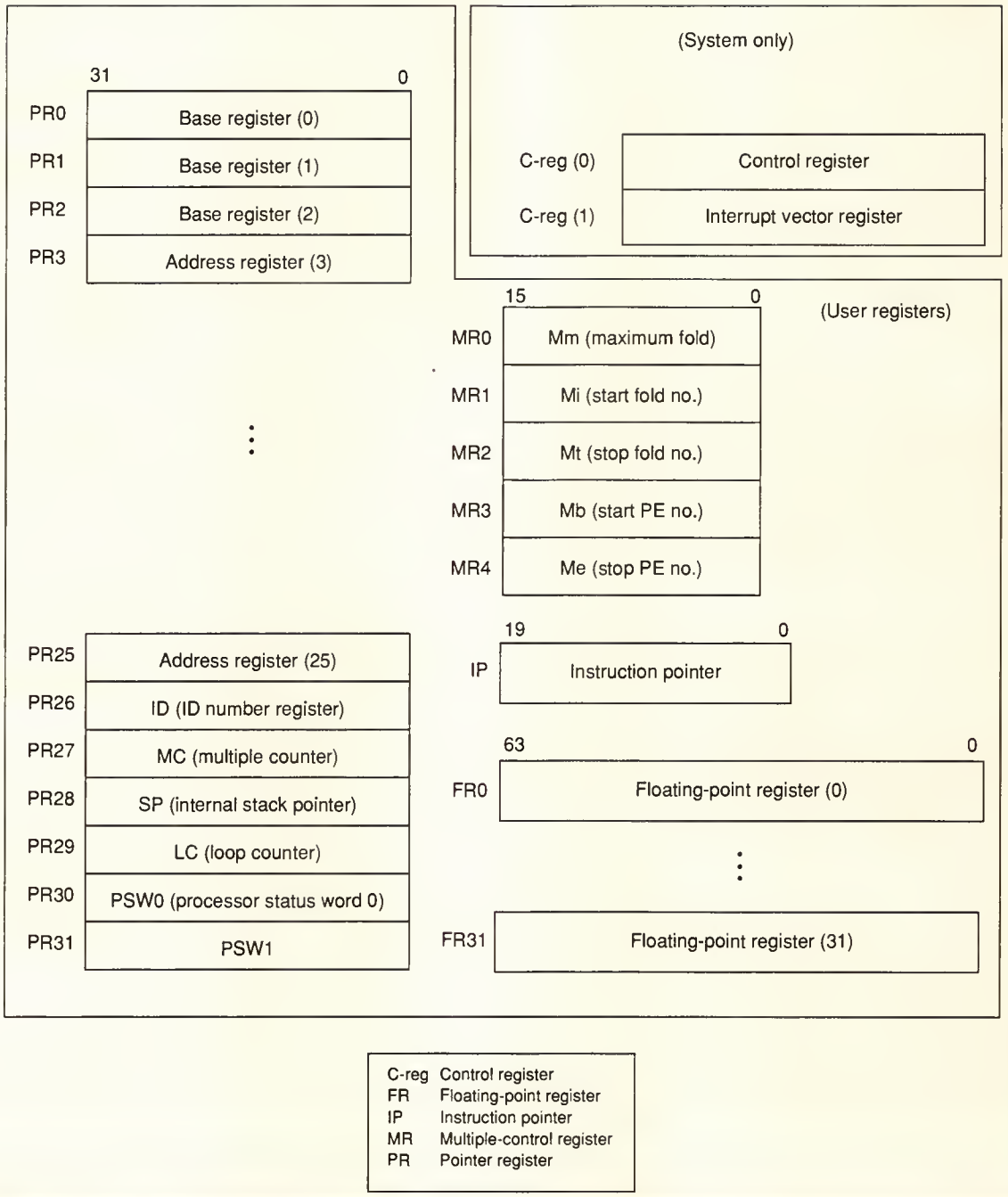


Figure 6. Programming model of the EPU.

systems. The system can access memories connected to the instruction bus and control registers of the processor from the data bus.

Figure 6 presents the EPU programming model. (See the task-switching section for an explanation of the M registers.)

The processor requires a 40-megahertz master clock and a 20-MHz slave clock for synchronization with the TCU. The internal machine cycle takes 50 nanoseconds. Performance peaks at 20 MIPS (million instructions per second) or 20 Mflops (million floating-point operations per second). A typical performance consists of a 7-Mflops fractal computation and a 4-Mflops execution of Gaussian elimination, not counting floating-store operations. Cycles per instruction average approximately 1.25, which corresponds to 15-17 MIPS.

Table 1.
Benchmark results.

LFK No.	Performance (Mflops) PE with cache (A)	PE without cache	Cray X-MP/48[15] (one processor) (B)	B/A (associated parallelizing index)
1	4.2	2.2	152.3	36
3	2.9	1.7	135.3	47
5	2.2	1.4	6.18	3
7	4.0	2.6	171.4	43
9	4.4	3.0	144.7	33
10	2.1	1.3	69.1	33
11	1.6	1.1	8.3	5
12	1.6	1.1	66.5	42

Table 1 contains the performance results of one PE for some Livermore Fortran Kernels (LFKs). We used an experimental Fortran compiler to generate object codes. The table includes values for one Cray X-MP/48[15] for reference. Performance ratios compare a parallel computer with the number of PEs in the column under B/A to the parallelizing index for one Cray supercomputer, assuming low overhead in the parallel processor. For example, the performance of 36 PEs equals that of one Cray in terms of LFK 1.

The most characteristic feature of EPU design is that the EPU processes double-precision floating-point instructions and other instructions in the same pipeline. In conventional devices, such instructions are usually dealt with through a microprogramming technique using a small amount of hardware. However, the EPU RISC architecture, an excellent circuit design technique, and simplified hardware produced a microprocessor with a fast, large operation unit. The EPU integrates 490,000 transistors—including ROM sites—into a 13.1 × 14.1-sq-millimeter die.

Implementing floating-point instructions at the same level as other instructions relieves code generation in compiler and processor control. An FPU that connects to a processor as a coprocessor usually creates considerable overhead due to external bus cycles or interrupt handling.

Other advantages accrue from a fast FPU. The most important one for this architecture is that fast floating-point division can occur without the existence of a dedicated divide unit. Through unique algorithm and hardware organization in the EPU, double-precision floating-point division can occur in eight internal cycles (400 ns in a 40-MHz operation) in latency or in seven cycles (350 ns) in pipeline mode. This approach also reduces computational cycles for most mathematical functions.

In a math coprocessor like the 68881 or 80387, square-root, exponential, logarithmic, and trigonometric functions are usually calculated by a Cordic algorithm. In Cordic, it takes n cycles to obtain a solution of n digits using a quasimultiplication operation. However, if a multiplier is available, one can apply a more efficient algorithm that uses a special series or an approximation formula.

Table 2 compares EPU cycle counts with those of other math coprocessors.[16-18] The internal machine cycle of each chip is pretty much the same. These comparisons are much more convincing when an application utilizes rare but important mathematical functions that are not supported in these coprocessors.

Table 2.
Comparison of cycle counts for the EPU and math coprocessors.

Functions*	EPU**	68882	R3010	μPD72691
Addition	2 (1)	26	2	8
Multiplication	2 (1)	46	5	11
Division	8 (7)	78	19	37
Sin	80†	320	111	120

* Double-precision operation
**In latency. () indicates pipeline mode.
† Average counts excluding register swapping.

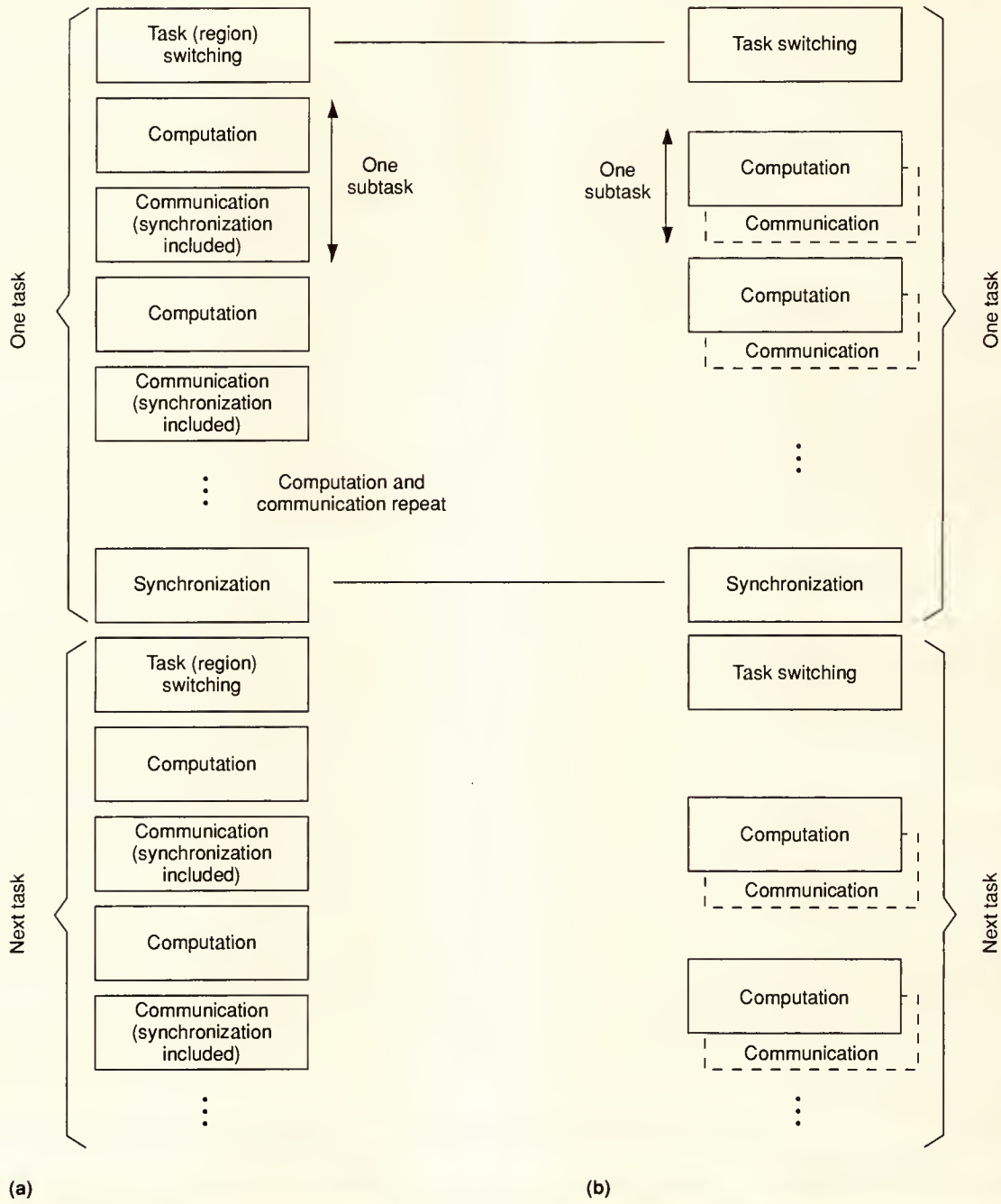


Figure 7. Comparison of operation flow in parallel computers for a conventional PE (a) and ADENA (b).

Reducing overhead

When an application program executes in a parallel computer, a consecutive sequence generally appears (see Figure 7). The underlying dashed boxes in Figure 7b indicate that the EPU and TPU are concurrently performing the adjoining tasks. Usually, a PE repeatedly executes a set of task switching, computation, communication, and synchronization operations. These combined tasks, which cause serious overhead costs for system performance, will continue to be essential and inevitable in the parallel processing environment unless a revolutionary algorithm or architecture is invented. In designing the EPU and TCU, we tried to

reduce the overhead as much as possible by adding the following functions.

Task switching. When a parallel processor deals with PDE problems, the grid size of the problem should be equal or smaller than the size of the PE array. When the grid size or grid lines exceed the number of PEs, each PE (real PE) turns into plural PEs (virtual PEs) in a multifolded arrangement (see Figure 8). Simulation space is partitioned into many sections that correspond to the structure of the PE array. The processor introduces a virtual network that connects virtual PEs. Figure 8 shows how the simulation is covered by a virtual network composed of $3 \times 3 \times 3$ real networks. The virtual PE array becomes a 3×3 real PE array. When a PE (PE*n* or PE*n*′) sends data to the virtual network, the real network functions three times (see the numbered arrows in Figure 8). Each PE is responsible for computing data along its grid lines, each of which corresponds to the PE's real position. Each PE calculates its virtual position and section number from its own PE number (or identification number) and current virtual position. The PE then compares its virtual position with the boundary conditions that the problem gives and switches to its next job.

The EPU provides instructions and dedicated hardware to deal with these operations. This hardware is valid when the PE is mapped into a linear or planner array. The multiple control unit shown in Figure 5 carries out these operations using MR0-MR4 and PR26 (Figure 6). Both PR27 and the flag bits in PR31 (processor status word 1) reflect the result. The PE can refer to the flags and judge the next job assignment within a few steps' execution.

Communication. Interprocessor communication is an essential process in the parallel processing environment. From the performance viewpoint, when the data acquired by a communication process passes to the next immediate process, the processor inevitably becomes idle. In ADENA, cooperation between the EPU and TCU relieves such overhead by overlapping communication with computation.

The TCU has a FIFO buffer that stores data to be sent to the network, an overflow counter that enumerates the number of data not acquired in the buffer, and a functional address generator. The EPU provides a special store instruction with a notification signal to the data bus, which requires that the TCU fetch data on the bus. The EPU executes this instruction when the stored datum is to be sent to the network. The TCU fetches or "snoops" the datum directly from the bus and sends it to the buffer. Data within the buffer passes to the network automatically. If the buffer is full, the notification signal counts up the overflow counter. Afterwards, when the buffer becomes available, the TCU fetches

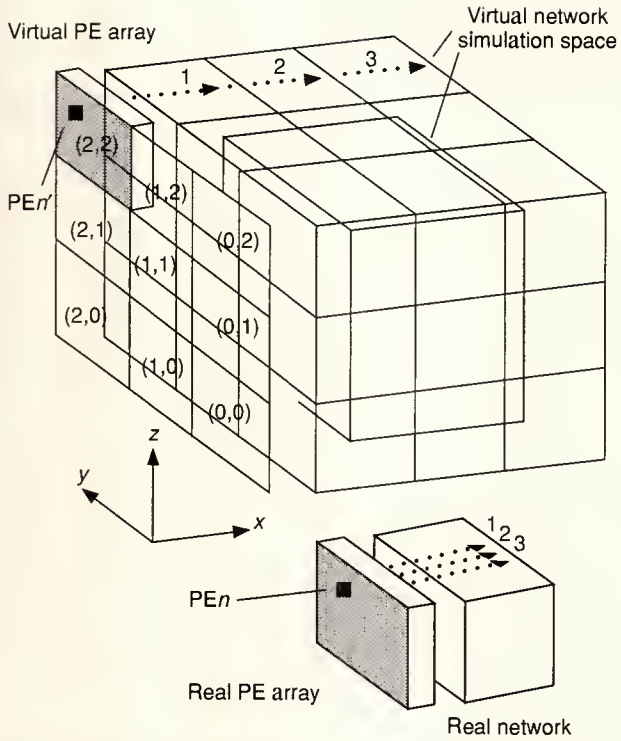


Figure 8. Multifolded PE array in a virtual and real network.

missed data from the LDM, while decrementing the counter. The address generator is functional enough to follow the same address sequence that the EPU produces to store the data to be transferred.

The TCU moves data generated in the EPU into the buffer or the network. The process is like an on-chip FIFO unit that extends to a part of external memory in which the address register in the EPU and the address generator of the TCU play the roles of write pointer and read pointer of virtual FIFO. From this point of view, we call this mechanism FIFO emulation. Thus, computation in the EPU successfully overlaps with data transmission in the TCU (see Figure 7b). The TCU receives data from the network and stores them to memory by cycle stealing. When the data generation in the EPU is not so frequent, substantial communication overhead equals the time period between the TCU fetching the last datum from the EPU and the arrival of this datum at the destination PE. The effectiveness of this mechanism, verified in several previous applications, provided a 10 to 30 percent leap in performance compared with a sequential execution of computing and communication.

Synchronization. This process occurs frequently when PEs switch to new subtasks, receive instructions

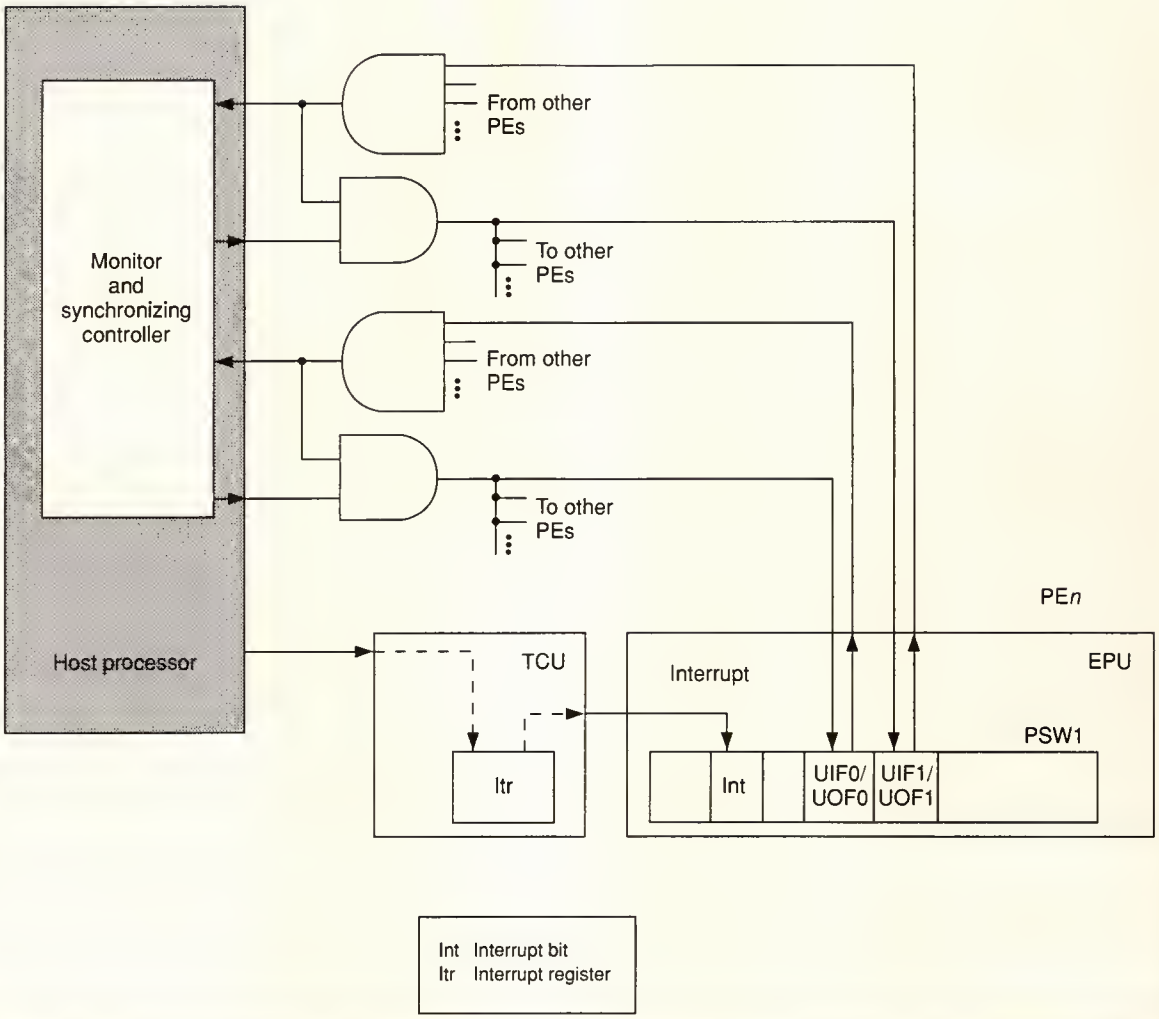


Figure 9. Synchronization facility.

from the host processor, or begin communication. In ADENA, an autonomous control sequence synchronizes events without host-processor control (see Figure 9). The EPU has four external pins, UIF0/1 (user input flag 0/1) and UOF0/1 (user output flag 0/1), which directly connect to the corresponding bits in PR31 (PSW1)—UIF0/UOF0 and UIF1/UOF1. These bits perform a double function. The EPU writes to UOF0 and UOF1 and reads from UIF0 and UIF1. Either one of two UIF/UOF pairs can send and receive a report on the synchronizing condition of each PE or all PEs. The other performs a write completion of individual PEs and all PEs. In other words, each PE writes the answer for a synchronizing condition to UOF0 and sets UOF1 at the same time, which signifies the completion of an individual synchronization process.

As all the PEs finish their own synchronizing processes, UIF1 is asserted. Each PE recognizes UIF1, checks UIF0, and begins processing a new task. This process corresponds to the statements as "if (all PEs) then" or "if (any PE) then." The interrupt register of the TCU, which handles interrupts to the EPU, contains interrupt requests from the host processor and information on ECC errors caused by LDM and network accesses. An interrupt signal from the TCU could deal with synchronizations accompanied by interprocessor communication or retrial of execution due to faults. This hardware is useful for forced synchronization by the host processor or making debugging tools.

The advantages of VLSI technology apply well to the field of parallel computing. In the ADENA computer, two VLSI chips form the organization of the PE. The resulting 42.6 sq-cm PE (including memories) reaches a peak performance of 20 Mflops

with a peak data-transfer rate of 40 Mbytes per second per PE.

In the 256-PE ADENA, 1.5-Gflops processing occurs in fractal computation in which operations are almost completely parallelized into all PEs. In ordinary PDE applications that include a great deal of synchronization, communication, and routines that include only a part of the PEs, we have observed levels of 300 to 400 Mflops. We expect this will be the case in other scientific/engineering applications. We estimate the cost of the machine will be about one tenth of that for a vector-type supercomputer. We believe that ADENA is the first parallel computer to achieve a Gflops-level of computation through massively parallel processing.

Acknowledgments

This work was a group effort. We express our appreciation to all our colleagues who assisted with this research project. We give special thanks to T. Okamoto, A. Wakatani, and S. Sasaki for performance analysis. We also thank T. Nogi and Y. Tanigawa for discussions and suggestions, and H. Mizuno, S. Horiuchi, and Y. Mano for encouragement.

References

1. K.E. Batcher, "STARAN Parallel Processor System Hardware," *Proc. 1974 NCC*, AFIPS Press, Reston, Va., Vol. 43, 1974, pp. 405-410.

2. K.E. Batcher, "Design of a Massively Parallel Processor," *IEEE Trans. Computers*, Vol. C-29, No. 9, Sept. 1980, pp. 836-840.

3. S.F. Reddaway, "DAP—A Distributed Array Processor," *Proc. First Ann. Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., 1974, pp. 61-65.

4. G.H. Barnes et al., "The ILLIAC IV Computer," *IEEE Trans. Computers*, Vol. C-17, No. 8, Aug. 1968, pp.746-757.

5. V. Zakharov, "Parallelism and Array Processing," *IEEE Trans. Computers*, Vol. C-33, No. 1, Jan. 1984, pp. 45-78.

6. T. Hoshino, "An Invitation to the World of PAX," *Computer*, Vol. 19, No. 5, May 1986, pp. 68-79.

7. J.P. Hayes et al., "A Microprocessor-Based Hypercube Supercomputer," *IEEE Micro*, Vol. 6, No. 5, Oct. 1986, pp. 6-17.

8. H. Kadota et al., "VLSI Parallel Computer with Data Transfer Network: ADENA," *ICPP Digest Papers*, Pennsylvania Univ. Press, University Park, Penn., Aug. 1989, pp. I-319-322.

9. K. Kaneko et al., "A VLSI RISC with 20-Mflops Peak, 64-Bit Floating-Point Unit," *IEEE J. Solid-State Circuits*, Vol. 24, No. 5, Oct. 1989, pp. 1331-1340.

10. Y. Nakakura et al., "A Versatile Data Transfer Controller for a Parallel Processor System," *VLSI Symp. Circuit Digest Tech. Papers*, IEEE Press, Piscataway, N.J., 1989, pp. 15-16.

11. K. Kaneko et al., "Network-Component Chip Set for a Parallel Processor System," *VLSI Symp. Circuit Digest Tech. Papers*, IEEE Press, 1988, pp. 39-40.

12. T. Nogi, "Parallel Machine ADINA, *Computing Methods in Applied Science and Engineering*, North Holland Press, Amsterdam, 1982, pp. 103-122.

13. T. Hoshino et al., "PACS: A Parallel Microprocessor Array for Scientific Calculations," *ACM Trans. Computer Systems*, Vol. 1, No. 3, Aug. 1983, pp. 195-221.

14. M. Annaratone et al., "Interprocessor Communication Speed and Performance in Distributed-Memory Parallel Processors," *Proc. 16th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, June 1989, pp. 315-324.

15. C. Eoyang et al., "The Birth of the Second Generation: The Hitachi S-820/80," *Proc. 1988 Supercomputing Conf.*, IEEE CS Press, Nov. 1988, pp. 296-303.

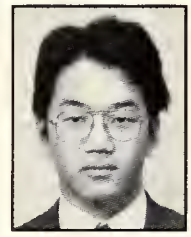
16. C. Rowen et al., "The MIPS R3010 Floating-Point Coprocessor," *IEEE Micro*, Vol. 8, No. 3, June 1988, pp. 53-62.

17. *MC68881/MC68882 Floating-Point Users Manual*, Prentice Hall, Englewood Cliffs, N.J., 1987.

18. T. Nakayama et al., "A 6.7-Mflops Floating-Point Coprocessor with Vector/Matrix Instructions," *IEEE J. Solid-State Circuits*, Vol. 24, No. 5, Oct. 1989, pp. 1324-1330.


Kaneko


Nakajima


Nakakura


Nishikawa


Okabayashi


Kadota

Katsuyuki Kaneko is a research engineer in the Semiconductor Research Center at Matsushita Electric Industrial Co., Ltd. He has been involved in the design and development of several microprocessors and peripherals. He is currently researching the architecture and organization of VLSI-oriented, highly parallel computer systems.

Kaneko received the BS and MS degrees in communication engineering from Tohoku University. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan.

Masaitsu Nakajima is a research engineer in the Semiconductor Research Center. He is currently investigating VLSI-oriented processor architectures and developing a 64-bit RISC for the parallel computer system.

Nakajima received the BS degree in electronic engineering from Tokyo Institute of Technology. He is a member of the IEICE.

Yasuhiro Nakakura is a research engineer in the Semiconductor Research Center. He has been involved in the design of a 64-bit microprocessor and network controller for the parallel computer system. He is currently designing a 64-bit microprocessor.

Nakakura received the BS and MS degrees in applied physics technology from Osaka University. He is a member of the IEICE.

Junji Nishikawa is a research engineer in the Semiconductor Research Center. He has been involved in the design of a 64-bit microprocessor.

Nishikawa received the BS and MS degrees in applied mathematics and physics from Kyoto University. He is a member of the Information Processing Society of Japan.

Ichiro Okabayashi is a research engineer in the Semiconductor Research Center. He has been involved in the design of 32-bit microprocessors. He is currently designing a network controller for the parallel computer system.

Okabayashi received the BS and MS degrees in electronic engineering from the Tokyo Institute of Technology. He is a member of the IEICE.

Hiroshi Kadota is a manager of the parallel computer ADENA development group in the Semiconductor Research Center. His interests include parallel processing. He had been involved in the development of two 16-bit MPUs and has designed an 8-kilobit content-addressable memory and a 32-bit MPU.

Kadota received the BS and MS degrees in electronic engineering from Kyoto University. He is a member of the IEICE.

Readers can direct questions concerning this article to Katsuyuki Kaneko, Semiconductor Research Center, Matsushita Electric Industrial Company, Ltd., Moriguchi, Osaka, 570, Japan.

IEEE MICRO

1990 and 1991 Editorial Calendar

JUNE 1990

Hot Chips, Part 2

Articles from the IEEE Computer Society TCMM's Hot Chips Symposium continue with insights into negotiations of the Endian Wars, 8087 family stack problems, TI's floating-point unit, Motorola's 68040, and an 88000 family update.

Ad closing date: May 1

AUGUST 1990

Communications and buses

Making RISCs work together; choosing the best software and interfaces; adding PC slaves and MS-DOS, memory technology, LANs, message-passing techniques...

Ad closing date: July 1

OCTOBER 1990

Digital Signal Processing

Is it the end of the supercomputer era? Comparison of supercomputer Linpack benchmark price/performance rankings with the latest 32-bit floating-point DSP chips makes serious analysts contemplate the future of the supercomputer industry.

Ad closing date: September 1

DECEMBER 1990

Special European and Near East issue

With 1992 just around the corner, US companies feel the pressure to meet new economic rules to compete for sales in Europe. What are European companies doing to meet US plans for common-market penetration?

Submit manuscripts by: 5-15-90
Ad closing date: November 1

FEBRUARY 1991

Microprocessors in Education
Special 10th Anniversary issue

Finding it difficult to illustrate and demonstrate new hardware to students? Experts show you some of the best techniques. In this anniversary issue, *IEEE Micro* also looks at industry prospects for the next 10 years.

Submit manuscripts by: 7-15-90
Ad closing date: January 1

APRIL 1991

Special Far East issue

The latest technology from Japan, Korea, Taiwan, and the Pacific Basin and current TRON Project offerings.

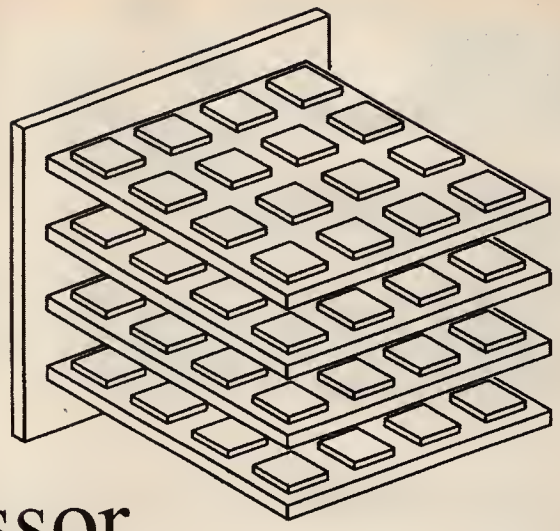
Submit manuscripts by: 9-15-90
Ad closing date: March 1

IEEE Micro helps designers and users of microprocessor and microcomputer systems explore the latest technologies to achieve business and research objectives. Feature articles in *IEEE Micro* reflect original works relating to the design, performance, or application of microprocessors and microcomputers.
All manuscripts are subject to a peer-review process consistent with professional-level technical publications. *IEEE Micro* is a bimonthly publication of the IEEE Computer Society.

Advertising information: Contact Marian Tibayan, Advertising Department, IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

Articles may change. Please contact editor to confirm.

FEATURE



A 4-Bit, 250-MIPS Processor Using Josephson Technology

An experimental 4-bit Josephson processor demonstrates the possibility of a Josephson computer system with a gigahertz clock. Constructed from 2,066 gates on a 5-mm × 5-mm die, it implements an eight-instruction set to enable the basic operations of digital signal processing. We introduce circuit techniques to suppress the cross talk from the AC power to the I/O lines and perspectives to improve the clock frequency.

Yuji Hatano
Shinichiro Yano
Hiroyuki Mori
Hiroji Yamada
Mikio Hirano
Ushio Kawabe

Hitachi Ltd.

Progress in the switching speed of submicrometer semiconductor devices is remarkable. However, even more overwhelming is the ultrafast switching operation of Josephson devices along with their extremely low power dissipation, even for relatively loose photolithographic technology. One Josephson device, the Pb-alloy (lead) version, demonstrated a lack of reliability and reproducibility that became a fatal drawback to its acceptance. However, the appearance of the Nb/AlOx/Nb (niobium/aluminum oxide/niobium) Josephson junction changed the situation decisively.

Now, many efforts continue toward the digital application of Josephson technology.[1-7] A big market exists for Josephson devices in large-scale supercomputers used for scientific purposes. Another most-attractive field for Josephson device application is that of digital signal processing. DSP requires a fixed sequence of operation such as multiplication and accumulation—just what Josephson devices do best.

Now that chip fabrication technology has advanced to the point where LSIs (large-scale ICs) with several thousand gates can be made, researchers need to construct a small-scale system for application in the near future. With such a system, researchers can extract information about the benefits and shortcomings of Josephson technology and determine what problems require solutions. This system should preferably be a one-chip processor because it would include all kinds of digital circuit elements.

Several systems have already been produced: a 1-Kbit static RAM (SRAM),[3] the data path of a 4-bit microprocessor,[1] its updated version with control ROM,[2] and four chip elements of a 4-bit microcomputer.[4-7] However, no one has yet produced a one-chip processor furnished with finite instruction sets. Therefore, we fabricated one that includes both a control path and a data path. It also includes all the basic DSP functions.

We plan to use this processor in a feasibility study of a GHz clock system, which has never been attained with semiconductor devices. Next, we will evaluate the performance of the Josephson circuit and extract information to improve the performance.

Here, we review the basic structure and operation principles of this Josephson device and describe the design and operation results of the new processor. We focus mainly on the circuit techniques required to realize a GHz clock in a Josephson processor. We also discuss the remaining problems in an effort to enhance the performance.

0272-1732/90/0400-0040$01.00 © 1990 IEEE

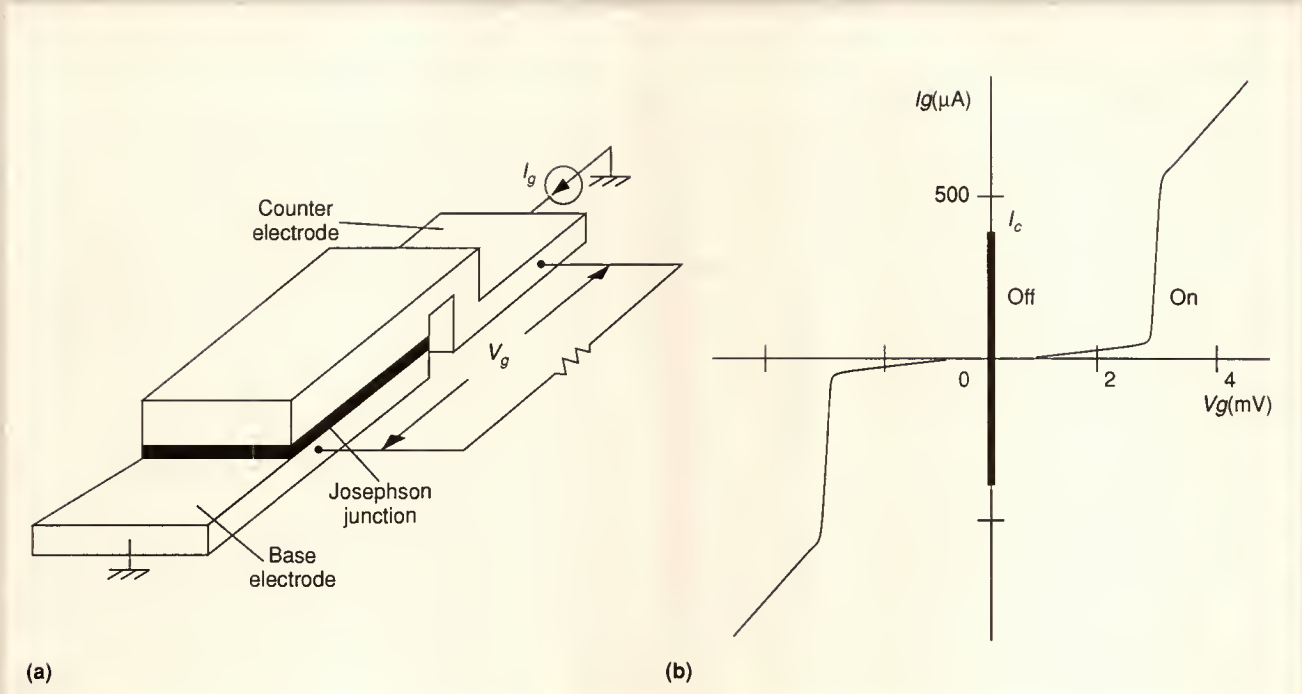


Figure 1. Basic structure (a) and characteristics (b) of the Josephson junction.

Josephson circuit basics

First, we briefly review the basic structure and operating principles of the Josephson circuit . Refer to Van Duzer and Turner[8] and Anacker et al.[9] for tutorial details.

Operating principles. At the liquid helium temperature (4.2 degrees Kelvin), some kinds of materials such as Pb or Nb show superconductivity. That is, current can flow in the material without resistance. Figure 1a shows that the Josephson junction is a sandwich structure of two superconducting electrodes with a thin insulating layer between them. If the insulating layer is thin enough, superconducting current flows through it in a tunneling effect. The current-voltage characteristics of the junction (Figure 1b) permit it to function in a superconducting state (off state) and a resistive state in which a few-mV voltage appears (on state). When the supplied gate current I_g exceeds the critical current I_c, the junction switches from the off to the on state. This switching ends in just a few picoseconds. The reverse on-to-off transition only happens when the gate current nears zero. This is the latching property of the Josephson device. In the on state, a region of constant voltage, called the gap voltage, exists. This region is characteristic of the electrode material; it measures 2.9 mV in Nb.

When we use the Josephson junction as a logic element, we can see that the input current is added to the gate current by transformer coupling or resistor coupling to exceed the I_c value. Thus, the transition from the off to the on state occurs. In the on state, the output current flows into the load resistor that is connected parallel to the junction and drives the next stage.

Power consumption. To avoid incorrect operation because of thermal noise, we must suppress the supplied gate current below the critical current with a finite distance. For example, the distance corresponding to the error rate of 10^{-14}/s is 20 μA. To permit a wide range of bias rate (I_g/I_c), we must ensure that I_c is greater than 200 μA. We determine the I_c level after considering the matching between the load resistor and the impedance of the transmission lines in the chip. Therefore, the Josephson device consumes power on the order of μW $[I_g(<\text{mA}) \times V_g(\sim\text{mV})]$.

Noise. Since the energy level of the Josephson device is low, it barely resists the noise coming through the I/O lines. What is worse, the impulse noise sometimes induces flux quantums to be trapped in the superconducting films. These trapped fluxes cannot be distinguished until the superconducting state is broken. They continue to generate a magnetic field and cause incorrect operation of the surrounding gates. Therefore, we must supply a low noise-interface apparatus between the Josephson circuit and the semiconductor circuits.

AC power supply. The latching nature of the Josephson device forces itself in general to be powered

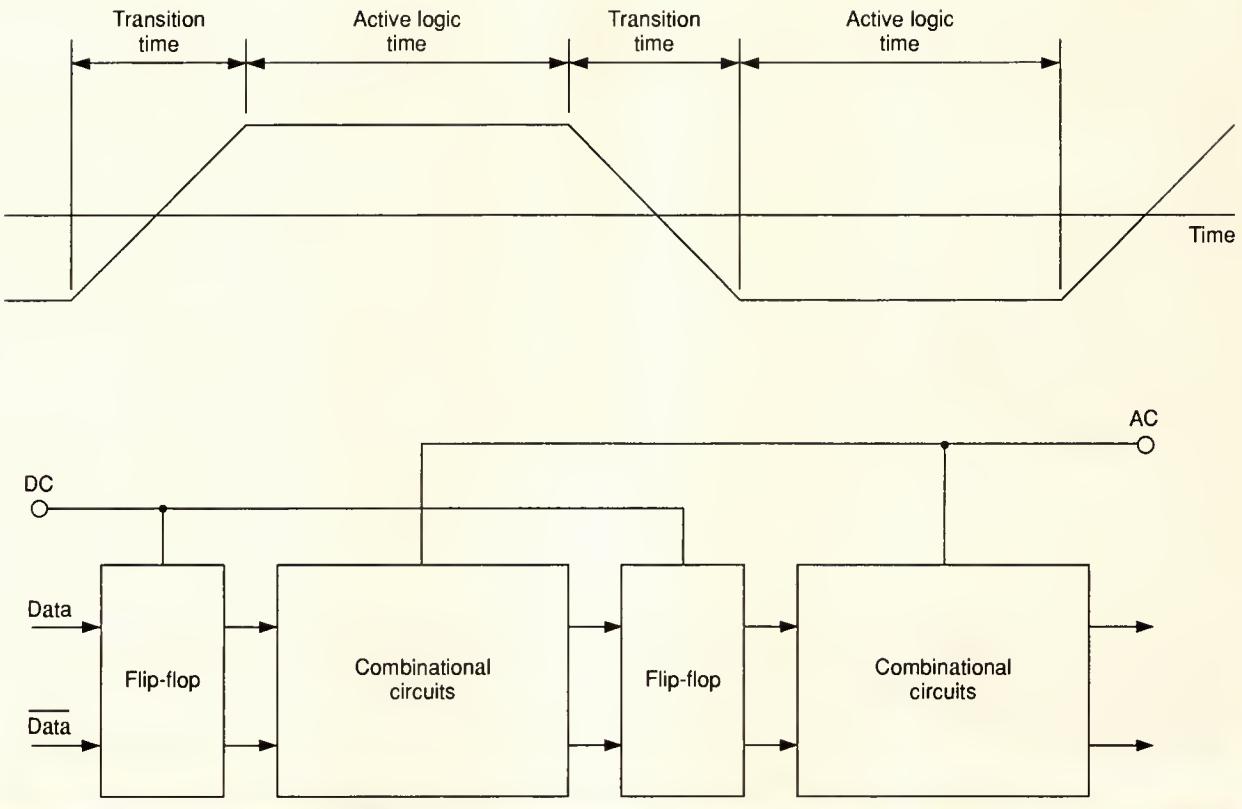


Figure 2. AC power scheme for a Josephson circuit.

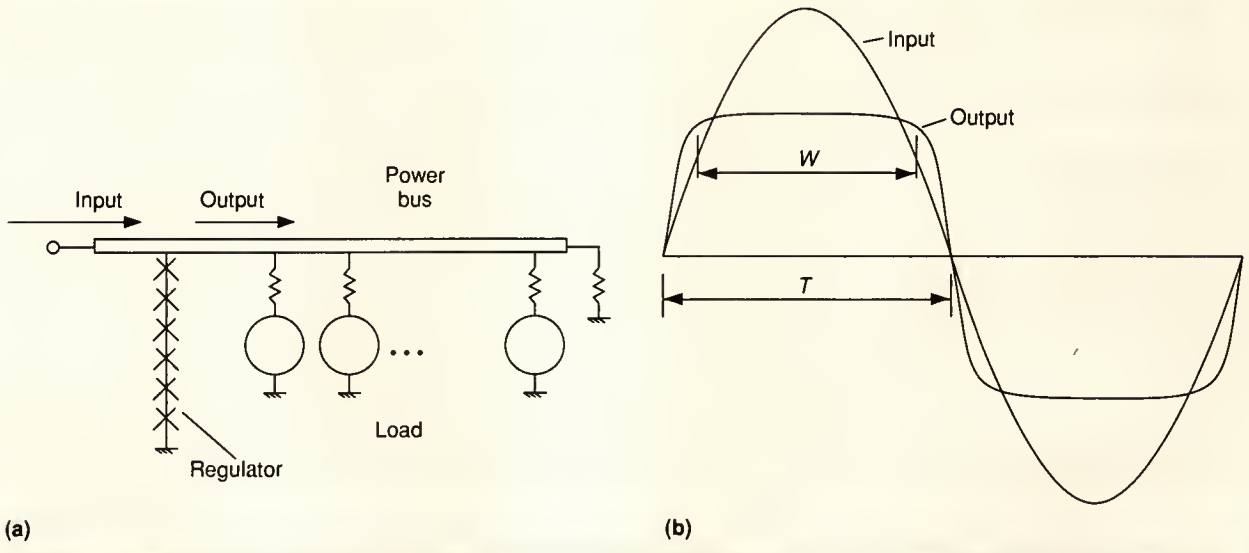
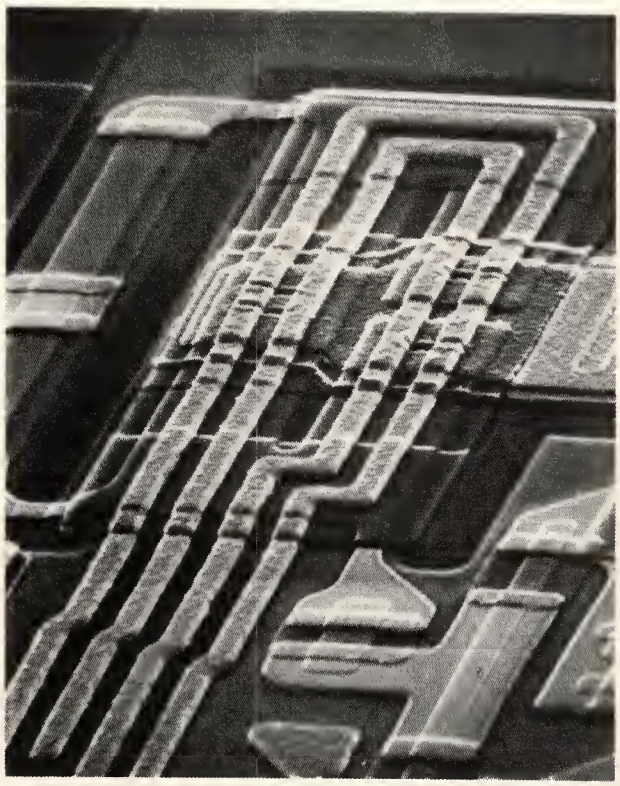
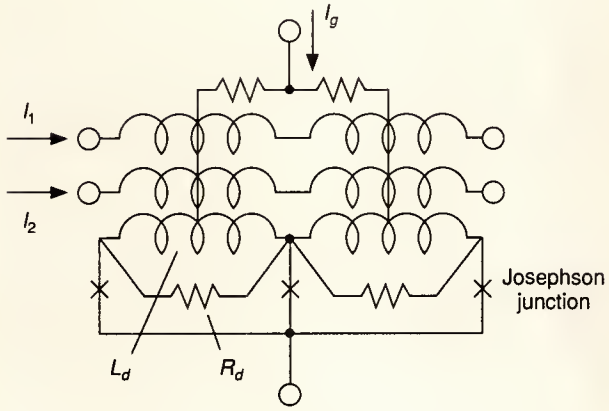


Figure 3. Construction (a) and operation (b) of the regulator. Duty = *W/T*.

with alternating current—one of the device's fundamental differences with semiconductor circuits. Flip-flops transfer data between AC cycles (Figure 2). The complementary signal line transmits the Not data (dual rail logic). Large, stacked Josephson junctions called regulators clip the bipolar sine wave into a trapezoidal one (Figure 3a,b) to supply a constant gate current to each gate.[10]



(a)



(b)

Figure 4. Construction of the three-junction interferometer gate: photomicrograph from a scanning electron microscope (a) and equivalent circuit diagram (b).

Table 1.
Layer construction of the Josephson circuit.

Layer	Material	Thickness (nm)	Function
M1	Nb	150	Ground plane
I1a	SiO	200	Ground plane isolation
I1b	SiO	150	Ground plane isolation
R	MoNx	95	Resistor
I1c	SiO	180	Resistor isolation
M2	Nb	160	Base electrode, first wiring
Barrier	Al, AlOx	5	Tunnel barrier
M3	Nb	80	Counter electrode
I2a	Si	160	Base electrode isolation
I2b	Si	380	Base electrode isolation
M3c	PbInAu	480	Counter electrode contact
I3	SiO	680	Base, counter electrode isolation
M4	PbInAu	1,400	Control electrode, second wiring
I4	SiO	2,000	Passivation

Cross talk. Each AC gate consumes about 300 μA in our design. Therefore, the total power current in a kilogate circuit attains 300 mA. Furthermore, the regulator consumes two times more AC current than the AC circuit itself to obtain an 80 percent duty ratio. This is the ratio of the active logic time W and the machine cycle time T shown in Figure 3. On the other hand, the I/O signal level is 400 μA, and its noise immunity is 20 percent at most. Therefore, the cross talk from the AC power current to the I/O lines should be less than – 80 dB. Furthermore, this cross-talk level must be attained at the clock frequency, which is 1 GHz in our target.

Care must be taken in packaging the circuit as well as in its construction to suppress cross talk. Cross talk results mainly from the ground impedance or the mutual inductance of the chip's bonding. Therefore, low-impedance bonding such as CCB (controlled collapsed bonding) works better than wire bonding. Josephson bonding must endure the thermal stress caused by the heat cycle between the cryogenic and room-temperature environments.

Gate construction. Figure 4a displays an actual transformer-coupled Josephson device constructed with a 2.5- μm design rule. Known as the three-junction interferometer,[11] this structure acts as a two-input Or gate and takes up an area of 90 μm × 45 μm. Its equivalent circuit diagram appears in Figure 4b. Table 1 lists the layer construction. The three-junction inter-

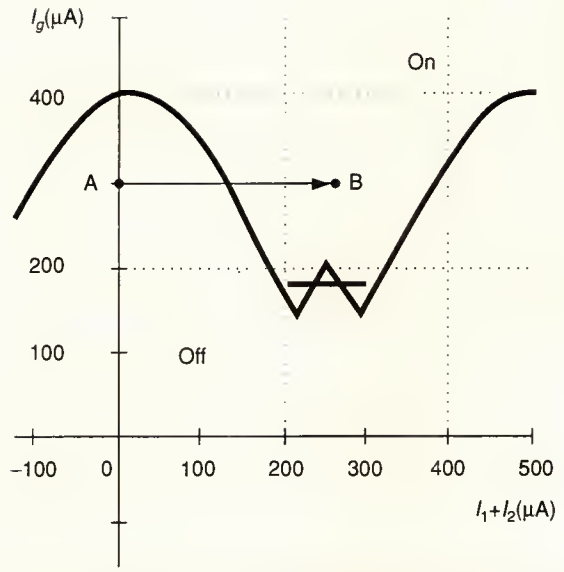


Figure 5. Threshold curve of the three-junction interferometer.

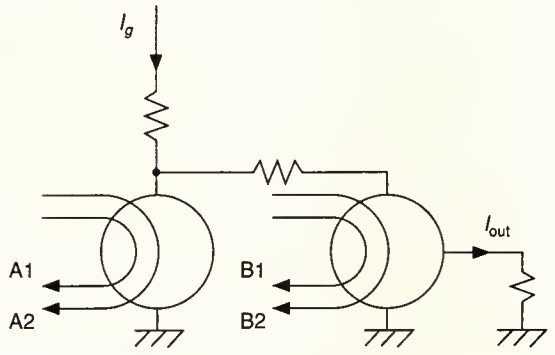


Figure 6. Wired And gate construction. (The three-junction interferometer appears as a circle.)

ferometer consists of three Josephson junctions with connecting device inductances L_d. The two input lines I_1 and I_2 are coupled by a transformer to an L_d. The dumping resistor R_d suppresses the resonance caused by the junction capacitance, and L_d realizes a smooth transition from the off to the on state. The output current flows into the input lines of the fan-out gates in series (serial fan-out), and an infinite number of fan-outs can be extracted.

The threshold curve of the three-junction interferometer can be seen in Figure 5. The inside curve reflects the off state, and the outside curve indicates the on state. The distinctive periodic structure results from the flux quantization in the superconducting loop constructed from the Josephson junction and the device inductance. We can set the bias point to point A by supplying the gate current beforehand. Applying the input current to transfer the bias point to B brings about the off-to-on transition.

The minimum delay time of the Or gate in Figure 4 is 9 ps. However, in a normal condition, that is, in the central bias point of the operating region, the delay amounts to 20 ps/gate. The And gate is the wired connection of the two three-junction interferometer devices, which are symbolized as circles in Figure 6. The output current appears only when both gates A and B have the input current. That is, we achieve the logic relation $I_{out} = (A1 + A2) \cdot (B1 + B2)$.

Conventional Josephson memory cells include at least one interferometer device. Therefore, the integration degree of the Josephson memory cannot be as great as that of the semiconductor devices with the same design rule. The physical size of the chip area limits the access time.

In addition, resistor-coupled gates[12-16] contain input signal currents that are directly injected into the junction through the coupling resistors. These gates produce less switching delay and occupy a smaller area because they do not contain an input transformer. They also use parallel fan-outs as do the usual semiconductor devices, though they limit the number of fan-outs to three or four.

Circuit design

The following section explains the circuit elements used in our Josephson processor and the I/O and power systems developed to suppress cross talk. These systems distinctively use the facts that the signal in the Josephson circuit is the current and that transformer coupling is possible.

DC flip-flop. From the circuit point of view the DC flip-flop, which realizes a DC power operation using latching Josephson devices, becomes the trickiest element. Many flip-flops have been proposed.[17-20] Figure 7 shows the modified Huffle circuit with a parallel resistor R_p.[20] The two three-junction interferometers G_1 and G_2 take the superconducting state and the voltage state complementarily according to the S and R inputs. The output current in the load inductance L_o changes between $+I_g$ and $-I_g$ correspondingly. If one of these two devices enters the voltage state, the other kicks back into the superconducting state by reducing the gate current to near zero value due to the current-keeping operation of L_o. R_p recovers the normal complementary operation after the irregular overlapping of the S and R inputs.

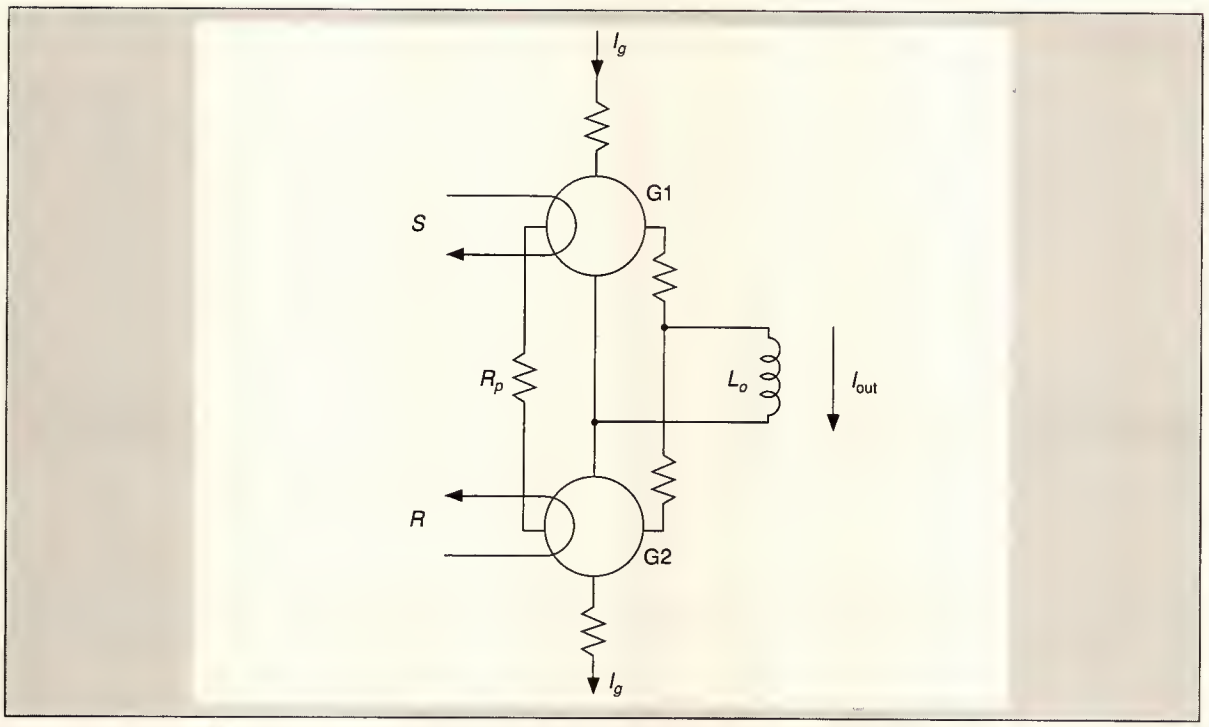


Figure 7. Huffle flip-flop with three-junction-interferometer gates.

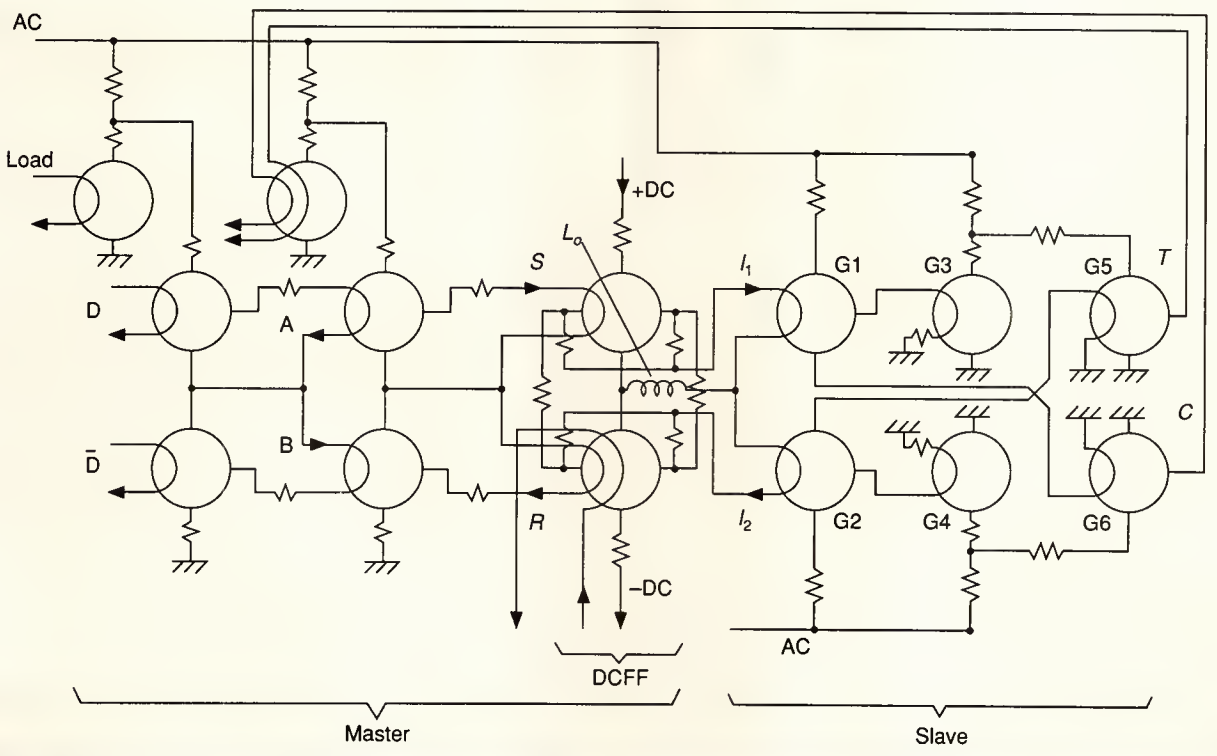


Figure 8. Construction of the master-slave flip-flop (D = data).

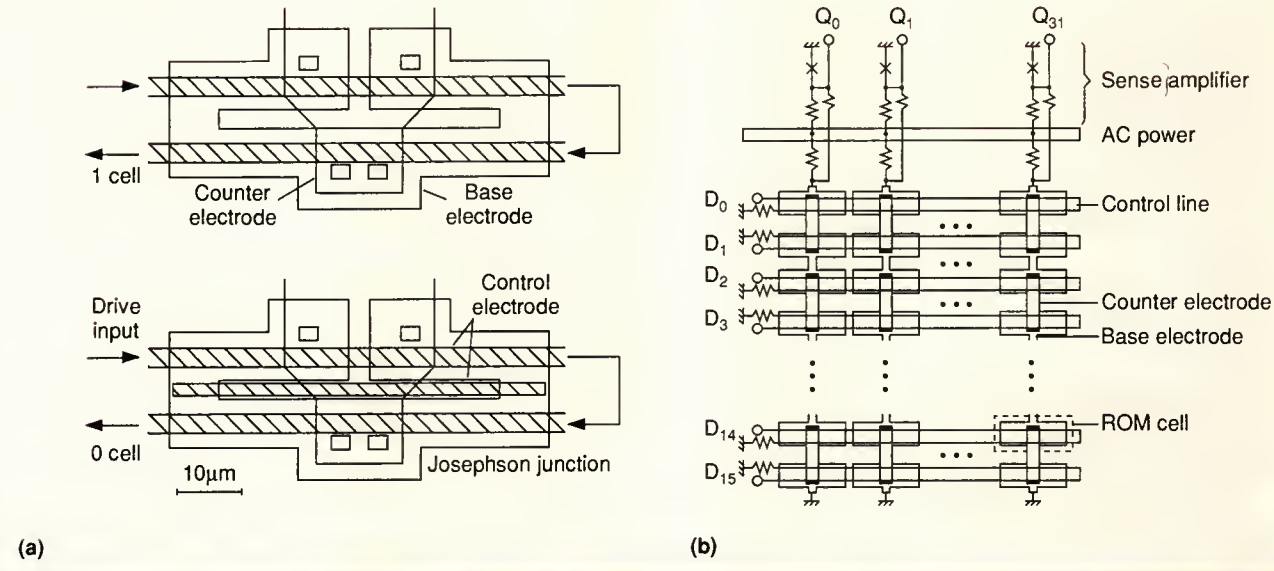


Figure 9. Construction of the ROM: a 1-bit element (a) and the block diagram (b).

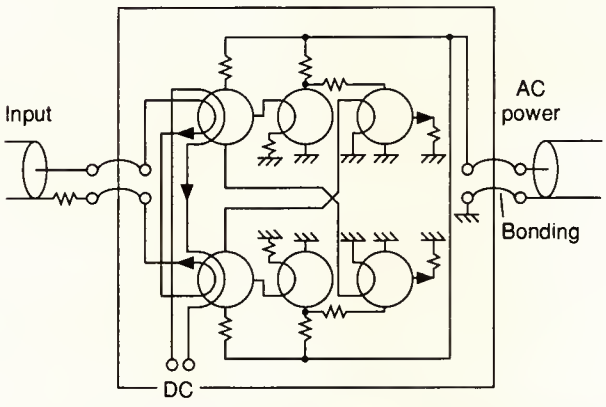


Figure 10. Construction of the input buffer.

Master-slave flip-flop. The master-slave flip-flop depicted in Figure 8 uses a DC flip-flop. The slave flip-flop reads the data of the DC flip-flop at the rising edge of the AC power cycle and keeps the data during the AC cycle. After T side gates (G_1, G_3, G_5) switch on, G_6 cannot switch even if I_2 switches on. The G_3/G_4 buffer stage realizes both the wide operating margin and the high output current. The master flip-flop prevents DC flip-flop data from being changed before the output of the slave flip-flop becomes fixed.

ROM. Figure 9a depicts the three-junction interferometer ROM cell. As shown in Figure 9b, the elements in one column are powered in common. One end of the column is grounded, the other end becomes the output terminal. If the input signal D_i appears, the output terminal Q_j reads the data of the i,j cell. The cell's 0 and 1 data can be selected by shorting or not shorting the counter and base electrodes of the cell with the control line electrode.

Input buffer. The input signal line couples with the transformer to the input buffer and terminates outside the chip, as shown in Figure 10. The ground potential of the AC circuit fluctuates greatly due to AC power current flowing through the finite ground impedance. The impedance results mainly from the bonding between the chip and the package. However, since the input signal line is electrically isolated from the AC circuit, this fluctuation does not get into the input lines. The cross talk caused by the mutual inductance of the bonding remains. However, since the input signal line and its return path to the termination outside the chip are set side by side, most of the cross talk is cancelled.

Output buffer. To separate the output signal lines from the AC circuit, we adopted a DC flip-flop as the output buffer. Two complementary signals S and R from the AC circuit drive the output buffer, as shown in Figure 11. The output node of one DC flip-flop gate connects to the pad, from which DC pulses with a voltage amplitude of about 2 mV can be obtained. We constructed the DC output buffer on a ground plane that is isolated from the AC circuit, and placed a 10-μm-wide ground plane trench between the AC circuit and the output buffer. The output of the AC circuit crosses

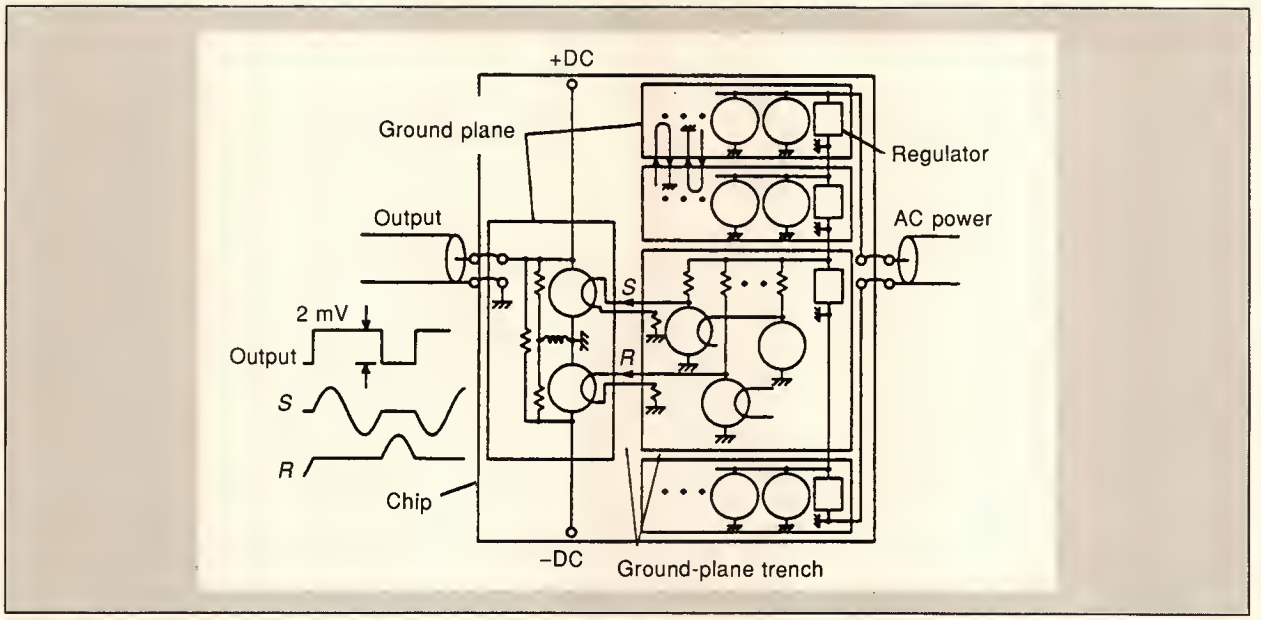


Figure 11. Construction of the DC output buffer and the stacked AC supply scheme.

the trench, magnetically couples with the output buffer, recrosses the trench at the back, and terminates at the AC ground plane. This ground separation reduces to one tenth the cross talk from the AC power current to the output lines.

The output buffer also plays another role. It transforms the signals from the bipolar, return-to-zero-format sine waves into the nonreturn-to-zero-format DC pulses to interface with the semiconductor circuits.

Stacked AC supply. We adopted a stacked AC supply to suppress the cross talk by reducing the absolute amount of power current. The AC circuit divides into several blocks according to their functions, as shown in Figure 11. We also apply the ground-plane separation techniques used at the DC output buffer here. A signal path starting in one block always terminates in the same block, after magnetically coupling the fan-out devices in the other blocks. Therefore, we keep electrical isolation, except at the regulator edges.

Each block contains about an equal number of powered gates. The numbers are completely balanced by the dummy resistors. Therefore, all blocks can use a common power current.

The signal path in the other blocks experiences three times the delay as the physical length, due to reflection in the worst case. The path lengths in the other blocks and the crossings over the ground-plane trenches should be minimized for delay suppression. Where a too-lengthy signal path in the other blocks is unavoidable, we insert buffer gates to separate the signal path.

As the AC circuit is divided into four blocks (Figure 11), we can decrease the AC current amplitude to one fourth, just as we can decrease the cross talk generated at the power terminal.

An on-chip transformer can also decrease the AC current amplitude.[10] However, use of an on-chip transformer severely limits the frequency bandwidth for operation to between several tens of MHz and several hundreds of MHz.

Processor construction

Using the above-mentioned circuits, we designed the data processor diagrammed in Figure 12 on next page. We constructed it from a 4-bit ALU, an 8-bit accumulator, a 16-word × 4-bit register file, a 16-row × 32-column ROM, and other components. We selected a horizontal microprogram because of its relatively high speed and ease of design. The register file stores both the data and the program. Although the data width is just 4 bits, the stored-program, full processor includes both the data path and the control path. All of the registers, including the program counter, are 4 bits wide.

The processor executes in two modes, programming and execution, which are selected by the Init input. In the programming mode the processor transfers data directly from the input terminal to the register file. In execution mode the processor sequentially executes the program written in the register file from the first address. The ALU performs five functions: Multiply, Add, Shift, Thru, and Complement. In addition, it

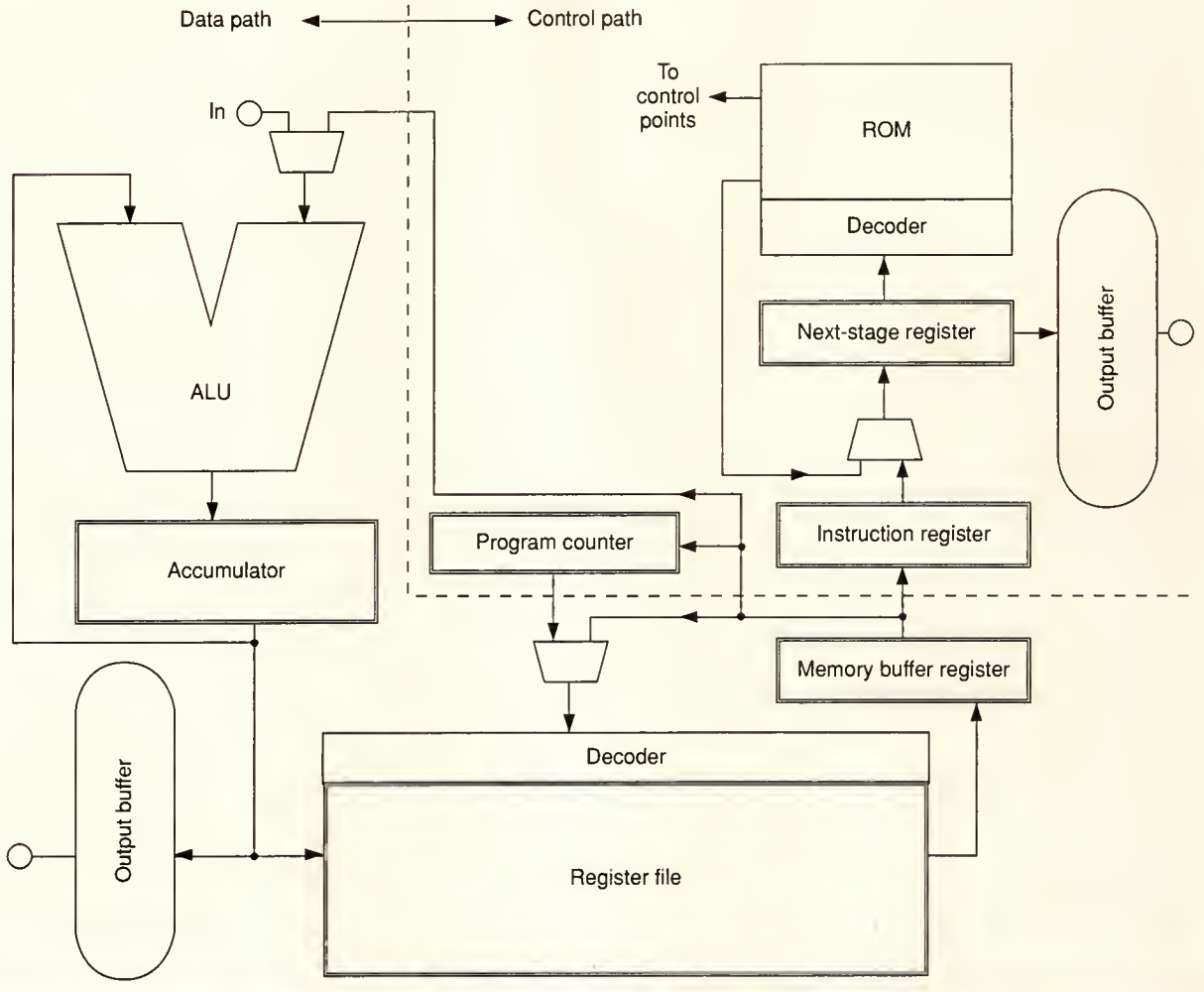


Figure 12. Block diagram of the processor.

implements Load, Store, and On-Condition-branch. Table 2 lists the operation code and functions for each instruction.

Figure 13 shows the internal state transitions for the instructions. Pipelining is not supported, and each instruction requires three to five cycles.

Table 3 lists the delay time of the longest path corresponding to each instruction. The average gate delay lasts 20 ps/stage, and the path delay lasts 9.0 ps/mm. One fan-out corresponds to a 165-μm increase in the output signal line. In addition, one And gate experiences twice the delay of one Or gate.

The longest path involves the store operation. Loop is not an execution command, but a path constructing a microcycle loop made of a next-state register (NSR), decoder (DEC), microprogram ROM, and multiplexer.

Less than 100 percent of the machine cycle time, which is the inverse of the clock frequency, can be used for the logic operation. At the time of AC power

Table 2.
Instruction set of the data processor.

Instruction	Opcode	Type	Function
Load *m*	0101	a*	Acc ← M(*m*)†
Store *m*	0110	a	Acc →M(*m*)†
BNLZ *m*	0111	a	If $Acc_3 = 1$, then $m \rightarrow PC$
Complement	1000	b**	−Acc → Acc
Thru *m*	1001	a	$m \rightarrow$ Acc
Sum	1010	b	Signal in + Acc → Acc
Multiply	1011	b	Signal in × Acc → Acc
Shift	1100	b	Acc, ~ $Acc_4 \rightarrow Acc_3$ ~ Acc_0

* Opcode: bits 7-4; address: bits 3-0
** Opcode: bits 3-0
† Register file with address *m*

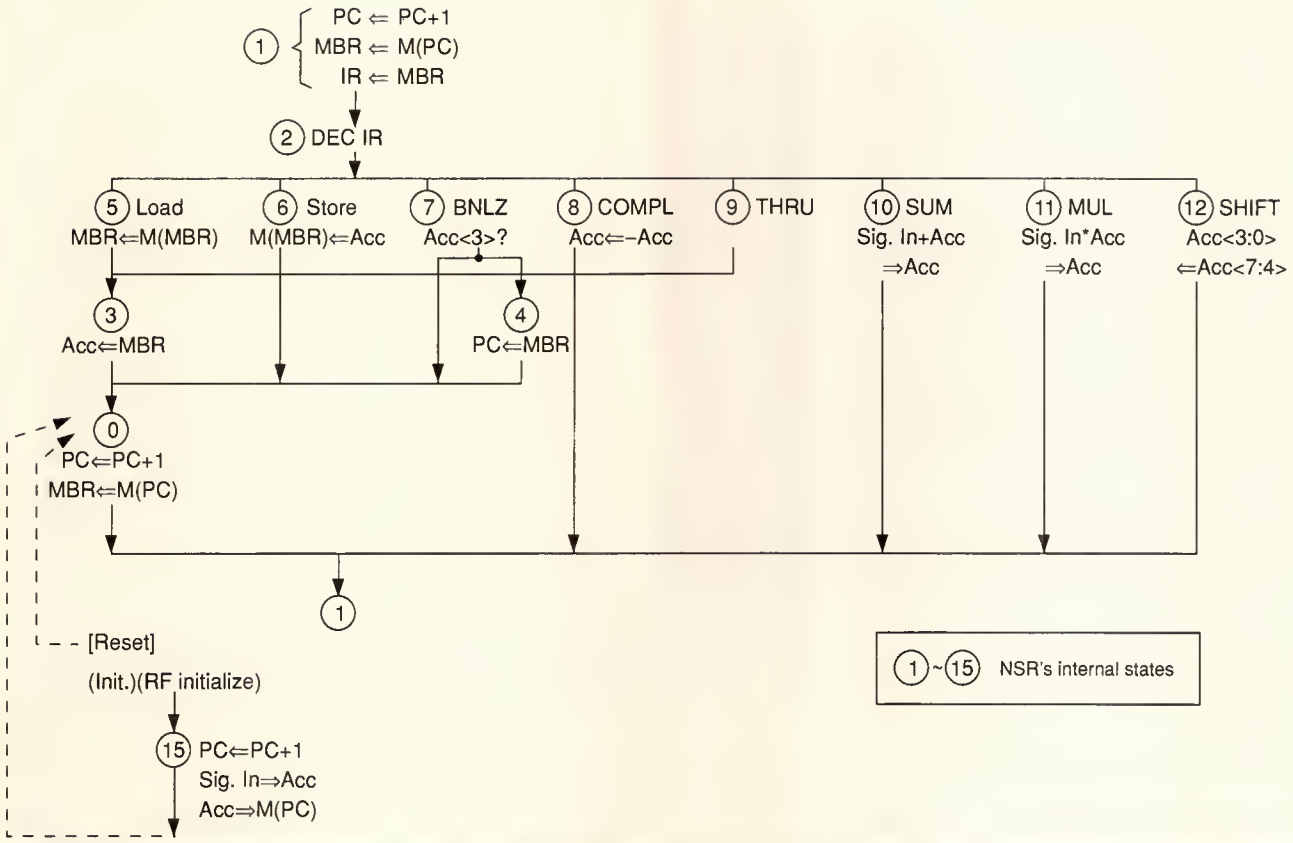


Figure 13. Internal state transitions of the processor.

transition, the logic operation stops. This transition time must not be too small, so as to avoid punchthrough. Punchthrough is a phenomenon that occurs when a gate that has risen into the resistive state cannot be returned to the superconducting state at the time of AC power transition.

Assume the AC power transition time τ equals 300 ps, the gate capacitance C_g equals 1.2 picofarads, the bias rate γ equals 75 percent, and the load resistor R equals 8 ohms. We can then set these in Harris and Chang's Formula 13[21] and achieve an estimated punchthrough probability P_o of 10^{-17}. This value cannot be neglected. If τ becomes 400 ps, P_o decreases to 10^{-26}. Considering the punchthrough phenomenon and selecting τ to be 300 ps, we obtain a critical path delay of 700 ps at a clock frequency of 1 GHz. The delay time in Table 3 shows that 1-GHz-clock operation is possible under a somewhat worse punchthrough probability. Since an average of four cycles are required for one instruction, this 1-GHz clock cycle corresponds to 250 MIPS.

Table 3.
Delay time required for each instruction. (Delay: 9.0 ps/mm; one fan-out: 165 μm; one gate: 20 ps.)

Instruction	Path length (mm)	Fan-out	Gate stages	Total delay
Store	36	31	17	710
Load	23	19	20	635
Multiply	20	16	21	624
Thru	20	17	16	525
(Loop)*	17	13	13	432

* Control-path loop constructed from the microprogram ROM, multiplexer, instruction decoder, and the next-state register.

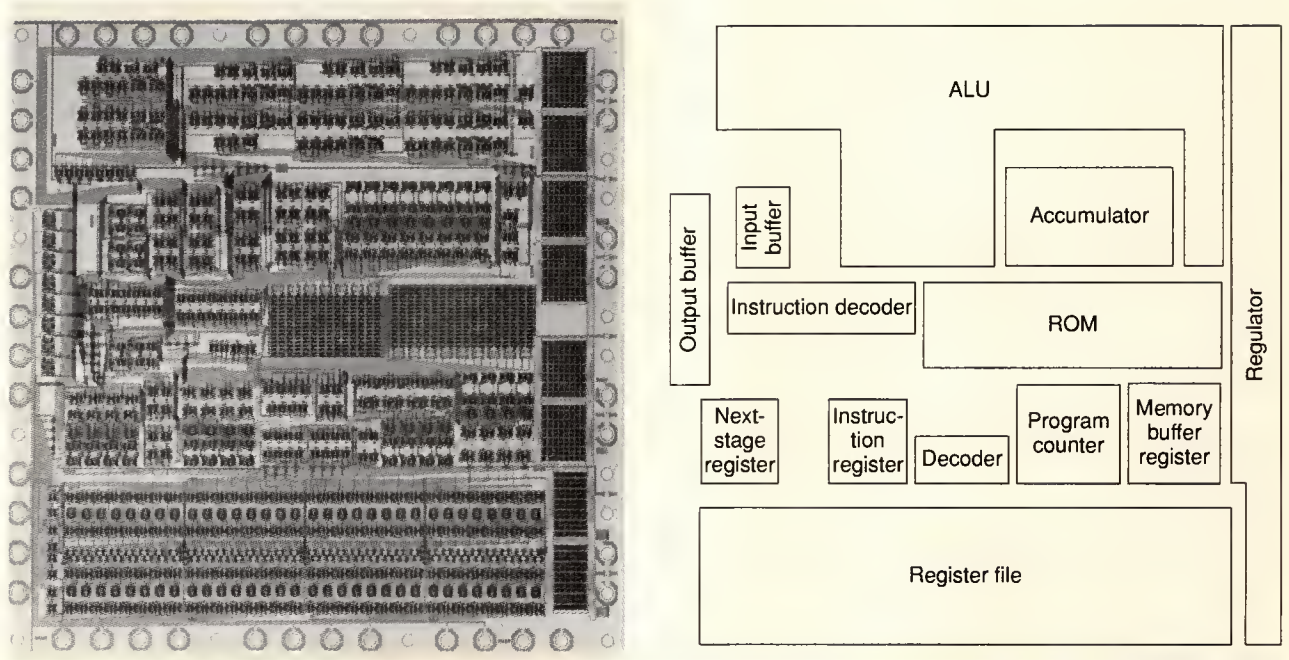


Figure 14. The 4-bit data processor.

Note that using the single-phase AC supply as shown in Figure 2 makes operation above 1 GHz difficult, although the single-phase supply was quite useful in easily realizing a 1-GHz operation. To achieve faster operation, we feel it important to use the multiphase approach, which utilizes two[4-7] or three[1,2] monopolar sine waves as the power source. However, this method requires strict phase matching in the package.

Figure 14 reproduces a photomicrograph of our chip. We fabricated it using the 2.5-μm Nb/AlOx/Nb process and included 2,066 three-junction interferometer gates and 9,027 resistors in a 5-mm × 5-mm area. The number of junctions totals 8,454, not including those used in the regulators. We incorporated six input lines: four Data, one Reset, and one Init. The Reset input clears the next-state register and the program counter. As shown in Figure 11, four blocks make up the AC circuit. The output lines are the most vulnerable to cross talk, so they appear on the other side of the power pads.

Experimental results

We measured the cross talk between terminals on opposite sides of a test chip to examine the effects of these design features. We mounted the chip on a package with 50-ohm strip lines and used wire bonding to connect the pads on the chip and the strip lines. We could not make the length of the bonding wire less than 1 mm because we had to have a slack wire to avoid wire destruction from thermal stress.

Without the ground separation, the cross talk amounts to about −30 dB at 500 MHz, as shown in Figure 15, curve A. This noise corresponds to a 1-GHz clock frequency for a bipolar power supply. The ground separation of the output buffer decreases the cross talk by about 20 dB, as shown in curve B. The stacked AC supply for four blocks reduces the cross talk to one fourth (curve C). The intense oscillation of the cross-talk curve results from resonance at the bonding wire. By using CCB techniques with the separated DC output buffer, we can decrease the cross talk to −80 dB in some areas of the frequency band, as shown in curve D. The clear difference between curves C and D shows the cross talk generated at the bonding wire.

However, we performed the following measurement using the wire-bonding technique, because the CCB process deteriorates the chip yield in the current stage.

Circuit operation. Figure 16 depicts a sample waveform from the ALU testing we performed in which we continuously applied the command format Add 0101. Each cycle normally adds five decimal values to the contents of the accumulator.

Examples of function test waveforms appear in Figure 17. We confirmed the internal state transitions shown in Figure 13 when we serially programmed the eight-instruction set. The waveforms in Figure 17a directly result from the AC-driven circuit, while we obtained those in Figure 17b through the DC output buffer. Apparently, the output buffer corrects the signal

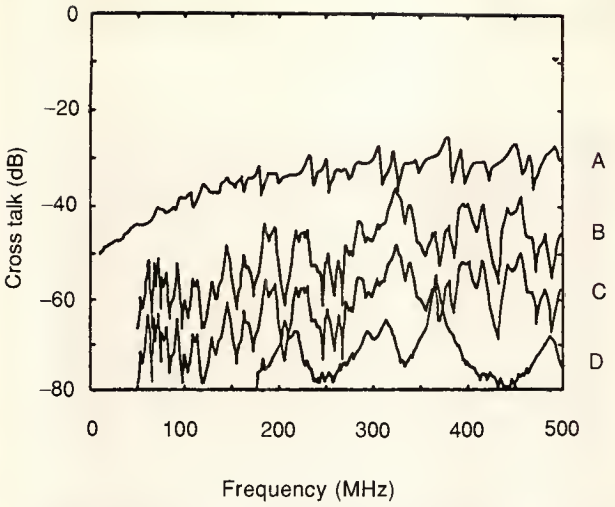


Figure 15. Cross talk from the AC power to the output line: Without DC output buffer or stacked AC supply and with wire bonding (curve A); without stacked AC supply, with DC output buffer and wire bonding (curve B); with DC output buffer, stacked AC supply, and wire bonding (curve C); and with DC output buffer, stacked AC supply, and CCB bonding (curve D).

Figure 16. The Add operation in the ALU.

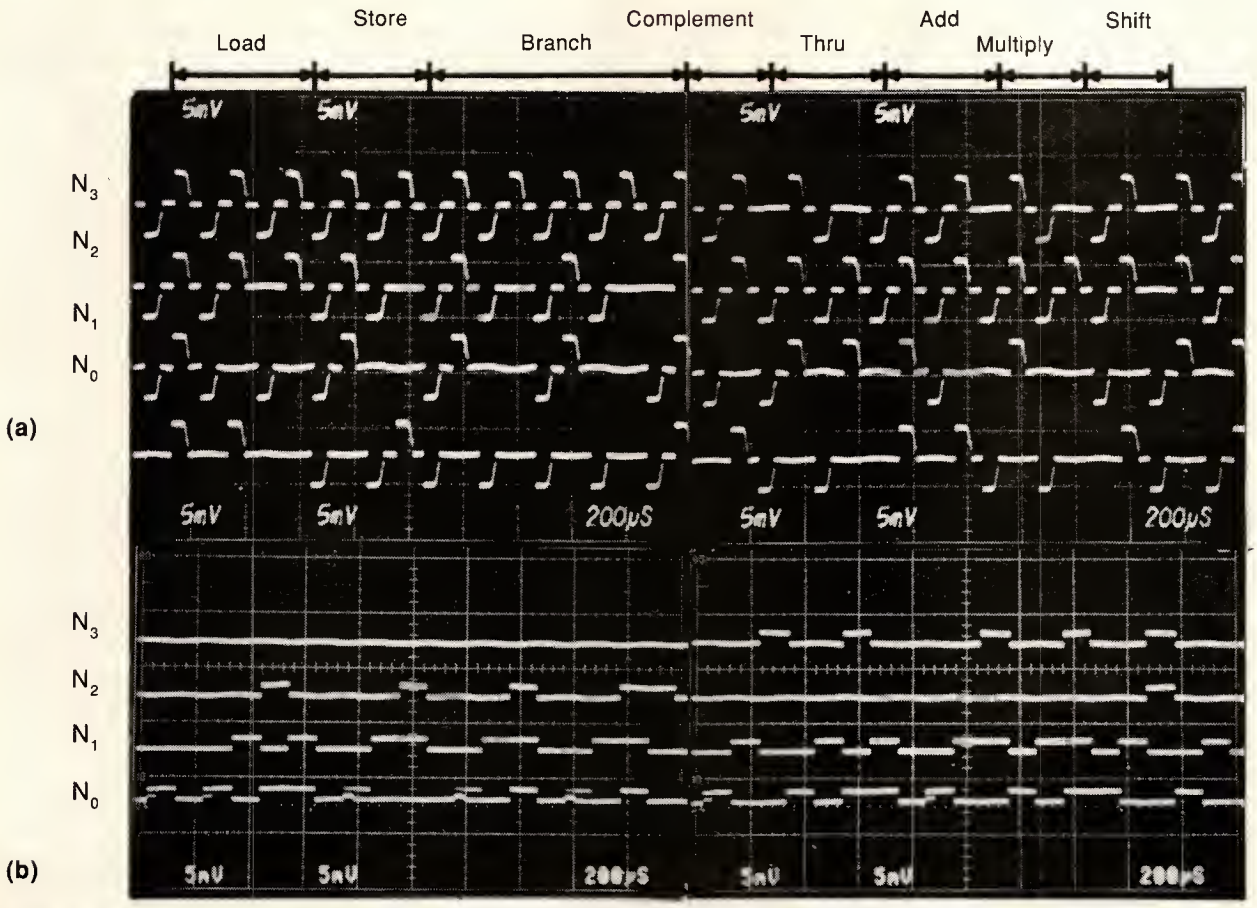
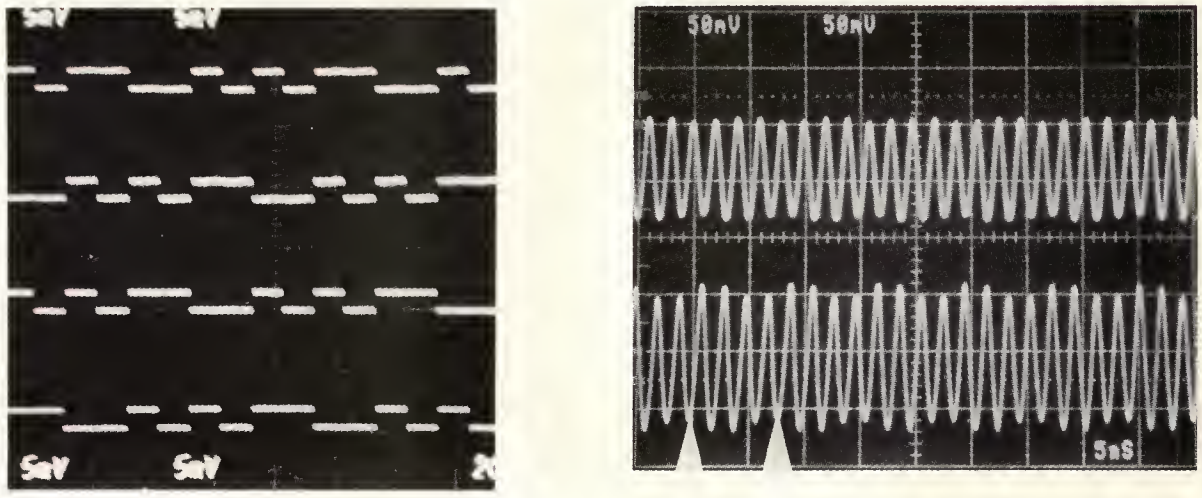


Figure 17. Internal state transitions corresponding to the instructions: Directly from the AC circuit (a) and output through the DC buffer (b). (N_i indicates the output of the next-state register.)



(a) (b)

Figure 18. Microcycle operation for high-speed testing: 10 KHz (a) and 1.02 GHz (b).

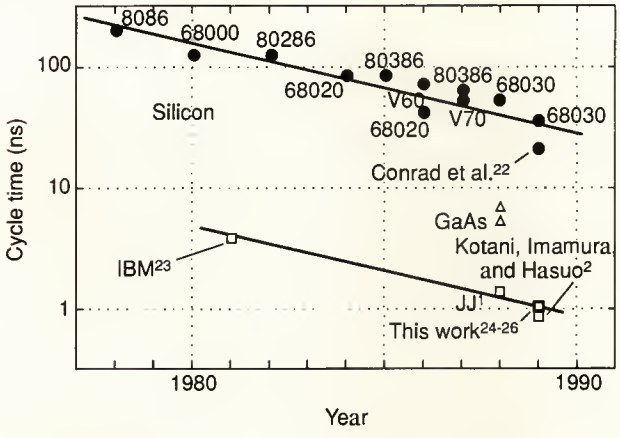


Figure 19. Development of microprocessors.

format to distinctive DC pulses.

We used primitive microdiagnostics for high-speed testing. We chose this method because providing GHz data pulses into a chip in a cryogenic environment and synchronizing them with AC power is difficult. The limited performance of available word generators makes it difficult. The Loop path in Table 3 shows a fixed pattern at the next-state register with a periodicity of eight clock cycles. This pattern occurs without inputting data pulses, except for single-phase AC power. At low frequencies, we can clearly examine the designed pattern, as shown in Figure 18a.

At high frequencies, cross talk at the wire bonding dominates, and we can no longer distinguish the data value. However, we can recognize operation as an artificial periodicity of four or eight clock cycles for an AC bipolar power supply. We can recognize this periodicity up to a power frequency of 510 MHz, or a 1.02-GHz clock frequency, as shown in Figure 18b. This frequency should not be limited by the circuit's own speed, but by the cross talk. The operating margin at a low frequency amounts to more than ± 30 percent for the regulator input current, but it decreases to nearly zero at 1 GHz due to cross talk. We've found it essential to use CCB techniques in achieving higher speed operation.

Discussions

Figure 19 indicates the recent developments in microprocessor cycle times. The fastest 32-bit CMOS microprocessor can operate at 50 MHz.[22] IBM achieved a 3.7-ns cycle time in a multichip cross-sectional model of a Josephson signal processor.[23] Kotani et al. reports 770-MHz[1] and 1.1-GHz operation[2] of a data path in a 4-bit microprocessor. The 1-ns cycle time obtained in our 4-bit experimental chip[24-26] and that in the Kotani data path[2] are 20 times faster than 32-bit CMOS microprocessors.

Let's estimate the performance of a Josephson 32-bit processor and that of the 4-bit one. The critical path of the ALU relates to the square root of the bit width. Therefore, the Josephson 32-bit processor will have a machine cycle of 700 ps × $\sqrt{8}$ + 300 ps, or 2,500 ps. We can see that 400-MHz operation is possible.

Of course, we would obtain this operation when using the 2.5-μm design rule. The path delay will linearly decrease as the rule decreases. Besides, most

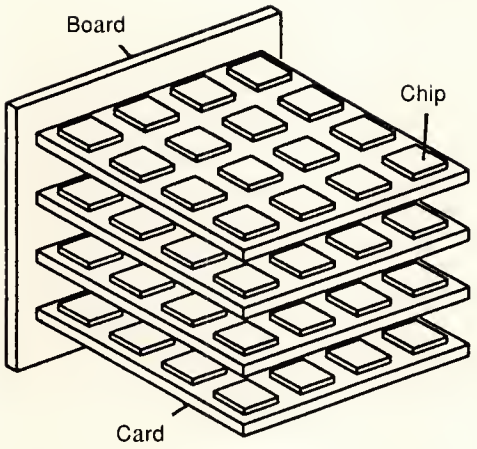


Figure 20. Packaging configuration of the future Josephson computer.

of the gate delay results from the CR time constant of the junction. This constant decreases as the square of the rule. Therefore, the Josephson 32-bit processor with a 1-μm design rule can be operated at 1 GHz.

The most advantageous application of the Josephson device might not be the general-purpose microprocessor. The huge liquid helium refrigerator required to realize a cryogenic environment would prohibit it.

Large-scale supercomputers, which seek the highest performance with a negligible cooling cost, create the best market for the high-speed nature of the Josephson device.

With 1-μm technology, 50,000 gates can be assembled on a 10-mm × 10-mm chip. The packaging techniques developed in an IBM project[9] that targeted the Josephson signal processor can be applied with few alterations. As shown in Figure 20, we will mount Josephson chips on a silicon substrate card that is mounted on a silicon substrate board. The superconducting wirings can connect the chips.

The degree of Josephson device integration is not superior to that of semiconductor gates with the same rule. On average, the Josephson gate area is three times greater than the ECL gates.

However, since the Josephson device consumes extremely low power and direct cooling with liquid helium is possible, the spacing between cards need not be large. Recent semiconductor packaging techniques enable the removal of a heat density of 100 W/sq cm or more. In contrast, the cooling limit of liquid helium is 1 W/sq cm. However, the power consumption of the Josephson device typically consumes power that is three orders of magnitude less than that of ECL devices. Moreover, it requires no space for a cooling water pipe between the cards. Therefore, the packaging density of a Josephson device in a finite space can be one order of magnitude higher than that of the semiconductor devices. This is an advantage when developing a high-speed clock.

Of course, some problems still await solutions. One, the so-called memory-neck problem, was one reason the IBM project stopped. Even with current technologies, the performance of the Josephson memory is not much better than that of the semiconductor ones. The access time of a Josephson 4-Kbit RAM with a 2.5-μm rule is 590 ps.[27] Performance of a 4-Kbit gallium arsenide RAM designed with 0.5-μm technology is 800 ps.[28] If the same layout rules are applied, the performance difference is larger. However, architectures designed to avoid the memory-neck property, such as the Connection Machine,[29] present one approach.

Magnetic-coupled gates are essential to construct flip-flops. However, in combinational circuits, the direct injection gate[12-16] is apparently preferable. The coexistence of magnetic-coupled gates and direct-coupled gates appears to be the most suitable approach.

Another problem regarding serial fan-out involves its application to the CAD tools used in semiconductor circuit design. Such tools only use parallel fan-outs and cannot be directly used for serial fan-out design. We manually routed the design of our processor. Note that Donath and Ogawa et al. have developed CAD tools for experimental use in treating serial fan-out.[30, 31] For large-scale integration design, the introduction of such tools is essential.

It is not easy to forecast the realization of the Josephson computer. If we consider only the technological point of view, we believe all of the techniques essential for a large-scale computer will be developed within several years. However, to obtain the performance advantages that can compensate for the disadvantage of cryogenic cooling, we realize many breakthroughs in circuitry, powering, and packaging techniques must occur. The Josephson computer might appear in the 21st century.

We've explained the design and experimental results of a 4-bit processor using Josephson technology. Although the number of instruction sets is just eight, this processor became the first stored-program Josephson processor containing both control and data paths. It includes 2,066 gates on a 5-mm × 5-mm die and performs the basic operations necessary for digital signal processing.

Our top priority was the realization of the GHz clock in the circuit design. We completely isolated I/O lines from the AC circuit and used a stacked AC power

supply to reduce the AC amplitude to one fourth of what it had been.

We tested functions at a low frequency. In addition, high-speed tests proved that the internal-state transitions of the microcycle occurred at clock frequencies up to 1.02 GHz.

Finally, we compared the processor's performance with that of recent semiconductor technologies and discussed the prospects for its improvement. AC power scheme, basic gate configurations, flux-trapping phenomena, as well as memory-neck property problems, still await solutions.

Acknowledgments

We conducted this work as part of the Large Scale Project of AIST/MITI's R&D of Scientific Computing System. NEDO (New Energy and Industrial Technology Development Organization) sponsored the project.

References

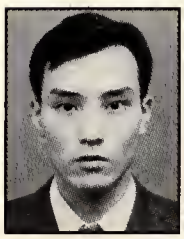
1. S. Kotani et al., "A Josephson 4b Microprocessor," *ISSCC Digest of Technical Papers,* Feb. 1988, pp. 150-151.
2. S. Kotani, T. Imamura, and S. Hasuo, "A Sub-ns Clock Josephson 4b Processor," *VLSI Symp. Digest of Technical Papers,* No. 3-2, May 1989, pp. 23-24.
3. Y. Wada et al., "A 570ps, 13mW Josephson 1kb RAM," *ISSCC Digest of Technical Papers,* Feb. 1988, pp. 84-85.
4. S. Kosaka et al., "Josephson Address Control Unit for a 4-Bit Microprocessor Prototype," *IEEE Trans. Magnetics,* Vol. MAG-25, Mar. 1989, pp. 789-794.
5. M. Aoyagi et al., "A Josephson 10-Bit Instruction ROM Unit for a Prototype Computer," *Extended Abstracts of 1989 Int'l Superconductivity Electronics Conf. (ISEC 89),* Vol. PL-20, June 1989, pp. 271-274.
6. I. Kurosawa et al., "A Data RAM Chip for a Josephson Computer Prototype," *Extended Abstracts of ISEC 89,* Vol. PL-31, June 1989, pp. 302-305.
7. H. Nakagawa et al., "A Josephson 4-Bit Processor for a Prototype Computer," *Extended Abstracts of ISEC 89,* Vol. DC-3, June 1989, pp. 387-390.
8. T. Van Duzer and C.W. Turner, *Principles of Superconductive Devices and Circuits,* Elsevier-North Holland, Inc., New York, 1979.
9. W. Anacker et al., "Josephson Computer Technology: An IBM Research Project," *IBM J. Research and Development,* Vol. 24, No. 2, Mar. 1980.
10. P.C. Arnett and D.J. Herrell, "Regulated AC Power for Josephson Interferometer Latching Logic Circuits," *IEEE Trans. Magnetics,* Vol. MAG-15, No. 1, Jan. 1979, pp. 554-557.
11. T.R. Gheewala, "A 30 ps Josephson Current Injection Logic (CIL)," *IEEE J. Solid-State Circuits,* Vol. SC-14, No. 5, Oct. 1979, pp. 783-793.
12. T.R. Gheewala and A. Muckerjee, "Josephson Direct Coupled Logic (DCL)," *Digest of Technical Papers, Int'l Electron Device Meeting,* 1979, pp. 482-482.
13. S. Takada, S. Kosaka, and H. Hayakawa, "Current Injection Logic Gate with Four Junctions," *Proc. 11th Conf. Solid State Devices, Japanese J. Appl. Phys. Suppl.,* Vol. 19, No. 1, 1980, pp. 607-611.
14. K. Hohkawa, M. Okada, and A. Ishida, "A Novel Current Injection Josephson Logic Gate with High Gain," *Appl. Phys. Lett.,* Vol. 39, 1981, pp. 653-655.
15. J. Sone, T. Yoshida, and H. Abe, "Resistor Coupled Josephson Logic," *Appl. Phys. Lett.,* Vol. 40, 1982, pp. 741-744.
16. N. Fujimaki et al., "9ps Gate Delay Josephson OR Gate with Modified Variable Threshold Logic," *Japanese J. Appl. Phys.,* Vol. 24, Jan. 1985, p. L1.
17. A. Davidson, "A Josephson Latch," *IEEE J. Solid-State Circuits,* Vol. SC-13, No. 5, Aug. 1978, pp. 583-590.
18. H.C. Jones and T.R. Gheewala, "AC Powered Josephson latch," *IEEE J. Solid-State Circuits,* Vol. SC-17, No. 6, Dec. 1982, pp. 1201-1210.
19. W. Baechtold, "A Flip-Flop and Logic Gate with Josephson Junctions," *ISSCC Digest of Technical Papers,* Feb., 1975, pp. 164-165.
20. A.F. Hebard et al., "A DC Powered Josephson Flip-Flop," *IEEE Trans. Magnetics,* Vol. MAG-15, Jan. 1979, pp. 408-411.
21. E.P. Harris and W.H. Chang, "Punchthrough in Josephson Logic Devices," *IEEE Trans. Magnetics,* Vol. AG-17, No. 1, Jan. 1981, pp. 603-606.
22. R. Conrad et al., "A 50 MIPS (peak) 32/64b Microprocessor," *ISSCC Digest of Technical Papers,* Feb. 1989, pp. 76-77.
23. M.B. Ketchen, D.J. Herrell, and C.J. Anderson, "Josephson Cross-Sectional Model Experiment," *J. Appl. Phys.,* Vol. 57 (7), No. 1, Apr. 1985, pp. 2550-2573.
24. Y. Hatano et al., "A 4b Josephson Data Processor Chip," *ISSCC Digest of Technical Papers,* Feb. 1989, pp. 234-235.
25. Y. Hatano et al., "A 4-Bit Josephson Data Processor Chip with DC Output Buffer," *Extended Abstracts of ISEC 89,* Vol. DC, No. 1, June 1989, pp. 375-380.
26. Y. Hatano et al., "A 4-Bit Josephson Data Processor Chip," *IEEE J. Solid-State Circuits,* Vol. SC-24, No. 5, Oct. 1989, pp. 1312-1316.
27. H. Suzuki et al., "A 4K Josephson Memory," *IEEE Trans. Magnetics,* Vol. MAG-25, No. 2, Mar. 1989, pp. 783-788.

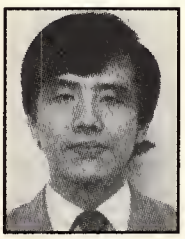
28. N. Matsunaga et al., "Half-μ Gate GaAs MESFET Technology Using Selectively Grown n+ Layer for High Speed Static RAM Fabrication," *Extended abstracts of IEEE GaAs IC Symp.*, 1989, pp. 147-150.

29. W.D. Hillis, *The Connection Machine*, MIT Press, Cambridge, Mass., 1985.

30. W.E. Donath, "Stand-Alone Wiring Program for Josephson Logic," *IBM J. Research and Development*, Vol. 24, No. 4, July 1980, pp. 480-485.

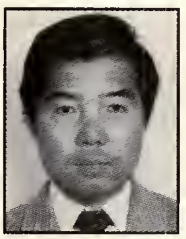
31. Y. Ogawa, H. Terai, and T. Kozawa, "Automatic Layout Procedures for Serial Routing Devices," *Proc. 25th Design Automation Conf.*, IEEE Computer Society, Los Alamitos, Calif., 1988, pp. 642-645.

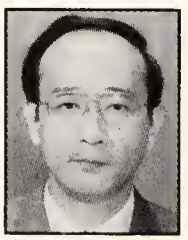

Hatano


Yano


Mori


Yamada


Hirano


Kawabe

Yuji Hatano is a researcher in the Central Research Laboratory at Hitachi Ltd., where he conducts research into Josephson devices. He received the BS and MS degrees in electronics engineering from Tokyo University. He is a member of the Japan Society of Applied Physics and the Institute of Electronics, Information, and Communication Engineers of Japan.

Shinichiro Yano also works as a researcher in the Laboratory, researching and developing Josephson integrated circuits. He holds BE and ME degrees in electronic engineering from Kyushu University, Fukuoka. He is also a member of the IEICE of Japan and the JSAP.

Hiroyuki Mori, a researcher at the Laboratory, has been engaged in research on Josephson junction devices since 1979. He studied electrical engineering at Nagasaki Technical High School and is a member of the IEICE of Japan and the JSAP.

Hiroji Yamada, a researcher at the Laboratory, works in process development for Josephson devices. Previously, he worked in the process development of magnetic recording, compound semiconductor light-emitters, and magnetic bubble-memory devices. He graduated from the Nippon Electronics College in electronic engineering and is a member of the JSAP.

Mikio Hirano, a senior researcher at the Laboratory, has been engaged in the research and development of electrical contact materials, bonding techniques for LSI, and Josephson devices. He received the PhD degree in electrical engineering from the Hokkaido University. He is also a member of the IEICE and the JSAP.

Ushio Kawabe, chief researcher of the Laboratory, serves as manager of the Superconducting Electronics Device Center at Hitachi. He has worked to develop single-crystal LaB cathodes for the electron microscope and investigated superconducting materials and the computer material design of high-temperature superconductors.

Kawabe received the MS and PhD degrees in physics from the Tokyo Institute of Technology. He is a member of the JSAP, the Institute of Electrical Engineers of Japan, and the IEEE Computer Society.

Questions concerning this article may be directed to Yuji Hatano at the Central Research Laboratory, Hitachi Ltd., 1-280 Higashi Koigakubo, Kokubunji Tokyo 185, Japan.

FEATURE

Realizing the V80 and Its System Support Functions

The 11-unit, 32-bit V80 microprocessor includes two 1-Kbyte cache memories and a branch prediction mechanism that is a new feature for microprocessors. We focus on the V80's pipeline processing and system support functions for multiprocessor and high-reliability systems. Without any performance drop, we realized multiprocessor and high-reliability systems using V80 support functions.

Hiroaki Kaneko
Nariko Suzuki
Hiroshi Wabuka
NEC Corporation

Koji Maemura
NEC IC Microcomputer Systems Ltd.

The 32-bit V80 microprocessor follows the V60 and V70 as the latest member of NEC Corporation's original microprocessor V-series family. Four years ago we developed the V60, the first commercially available 32-bit microprocessor in Japan.[1,2] Subsequently, we've seen the application field of 32-bit MPUs enlarged to include embedded applications as well as office automation equipment. Because of this growth, new requirements have arisen.

The control equipment field, which is the main target area of the V series, requires high performance (over 10 million instructions per second), real-time performance, and high reliability to build nonstop systems. These requirements demand a 32-bit MPU that uses new memory technologies such as nibble access or page access in dynamic RAMs. Normally, to decrease system cost, peripheral functions such as a math coprocessor or cache memory would be integrated on a silicon die as part of the MPU. At the beginning of the V80 development however, semiconductor manufacturing technology using submicron design rules promised to squeeze about one million transistors on a die.

To respond to the market needs, we targeted the V80 as the second-generation 32-bit MPU in the V series. We kept the architecture of the V80 the same as the earlier V60 and V70 to maintain programmability. An entirely new design and introduction of new techniques to achieve well-balanced pipeline processing provide 16.5-MIPS maximum performance at 33-MHz operation. Performance is almost three times higher than the 6.6-MIPS operation of the V70. When tested against the Dhrystone benchmark, the V80 at 33 MHz achieves a performance that is 4.2 times higher than the V70 (at 20 MHz).

The V80 contains 11 internal units, including separate 1-Kbyte instruction and data cache memories, and achieves seven-stage pipeline processing at 33 MHz. It contains 980,000 transistors in a 14.49-mm × 15.47-mm die fabricated by using a 0.8-μm, double-metal-layer CMOS (complementary metal-oxide semi-conductor) process.

Here we discuss the internal hardware structure of the V80, its pipeline operation, and its system support functions. Table 1 lists the main specifications of the V80, and Figure 1 reproduces a microphotograph of the chip.

Architecture overview

The programming architecture of the V80 (such as the instruction set, programmer register set, and virtual memory management) remains the same as that in the V60 and V70 32-bit MPUs. The single architecture of the

0272-1732/90/0400-0056$01.00 © 1990 IEEE

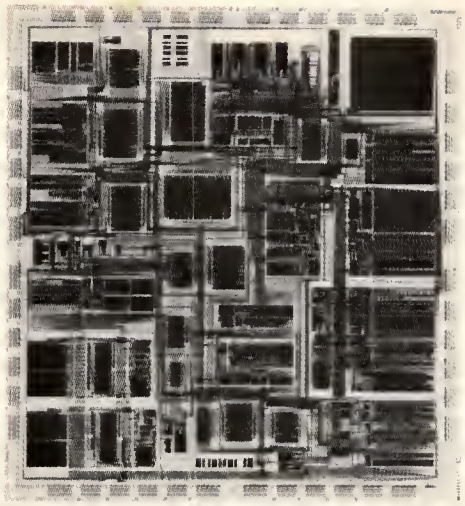


Figure 1. Microphotograph of the V80 chip.

V-series microprocessors ensures compatibility in the object code. We adhered to the basic concept of offering high programmability with generality and flexibility. The architecture contains:

- thirty-two 32-bit, general-purpose registers;
- 119 symmetrical types of instructions in the instruction set (four instructions are added to the V80);
- 21 orthogonal addressing modes and 14 data types; and
- a 4-Gbyte task space with a demand paging-based virtual memory system.

The single architecture lets users of the different MPUs use the same software without change. Users can apply the system software directly on the V80.[3] This software includes a real-time operating system (the RX616) that employs the TRON project's ITRON (industrial) specification, a real-time Unix operating system (the RX-UX832), and C compilers developed for the former V60/V70. The RX-UX832 real-time operating system, also based on the ITRON specification, contains the additional man-machine interface of the Unix System V, release 2.

The V80 contains some functional additions and improvements over the V70. These have no influence on the user's program and should be considered in limited system software such as an operating system. Most alterations exist in the privileged register set, as shown in Figure 2 on the next page.

We added a cache control word (CHCW) to control on-chip cache memories, a branch prediction mechanism, and a translation look-aside buffer, or TLB.

The earlier V70 contained some support functions for debugging a highly pipelined MPU with virtual memory management. The main function, the address trap, detects when a memory access occurs in a specified range and activates a trap. We changed some debugging specifications in the V80 to adopt new user requirements.

Applications that demand high reliability such as a nonstop system require that MPUs have support functions. Implementation of these functions by external circuits leads to an expensive system. The V70 contained functional redundancy monitoring (FRM) and bus error-exception functions to ensure high reliability.[4,5] In the V80 we added parity generation and error detection functions for address and data buses.

Table 1.
The V80 specification.

Item	Specification
Product name	μPD70832
General-purpose registers	32 bits × 32
Instruction set	V60/V70 upward compatibility
Virtual memory management	
Virtual memory space	4 Gbytes by demand paging
Real memory space	4 Gbytes
On-chip floating-point operations	32/64 bits, based on IEEE Std. 754
On-chip cache memory	
Instruction cache	1 Kbyte
Data cache	1 Kbyte
Monitoring function	By real address
Bus cycle (mode)	
V70-compatible	Two clock cycles
Advanced address	Output pipelined address
Burst	Five clock cycles (for 16 bytes)
High-reliability supports	Bus error trap/retry Function redundancy monitoring Parity check/error detection
Number of transistors	980,000
Process technology	0.8-μm, double-metal-layer CMOS
Die size	14.49 mm × 15.47 mm
Packaging	280-pin PGA with cooling fin
Operation clock frequency	25/33 MHz
Maximum performance	12.5/16.5 MIPS (at 25/33 MHz)

Overall design considerations

At the beginning of the V80 development, we targeted a two- to four-times higher performance than the V70 provided at the same clock rate. To gain this performance level, we adopted a new internal structure and effective techniques. After analyzing trace data from the V70, we discussed and used:

1) a sophisticated pipeline structure to execute basic

instructions in two clock cycles independently of the addressing mode,

2) separated on-chip cache memories, and

3) a branch prediction mechanism to control the execution of predicted branch instructions.

The basic design concept for the V80 was to implement the techniques that have been employed in sophisticated general-purpose computers.

We considered performance with real users in mind. For them, the high performance of application programs proved to be more important than the results obtained from testing against small benchmarks. The application-oriented instructions such as character-string manipulation, bit-string manipulation (bit build), and floating-point arithmetic instructions needed to be improved with dedicated hardware. To decrease the virtual memory management overhead, we reinforced the capacity of the TLB and employed a fully hardware-controlled replacement method.

We achieved the target performance with certain main techniques:

1) pipelined processing,
2) on-chip primary cache memory, and
3) high-speed branch prediction processing.

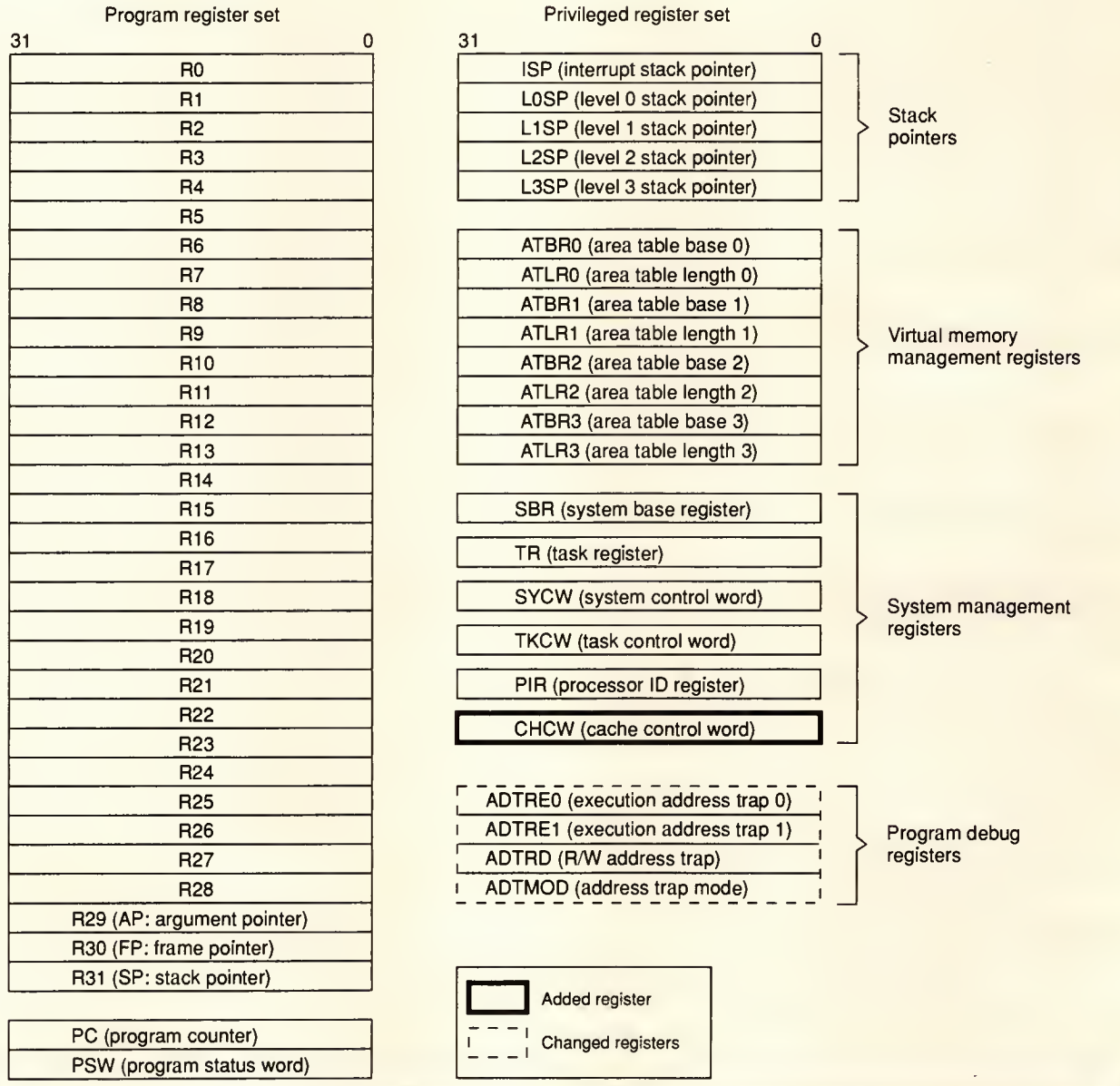


Figure 2. The V80 register set.

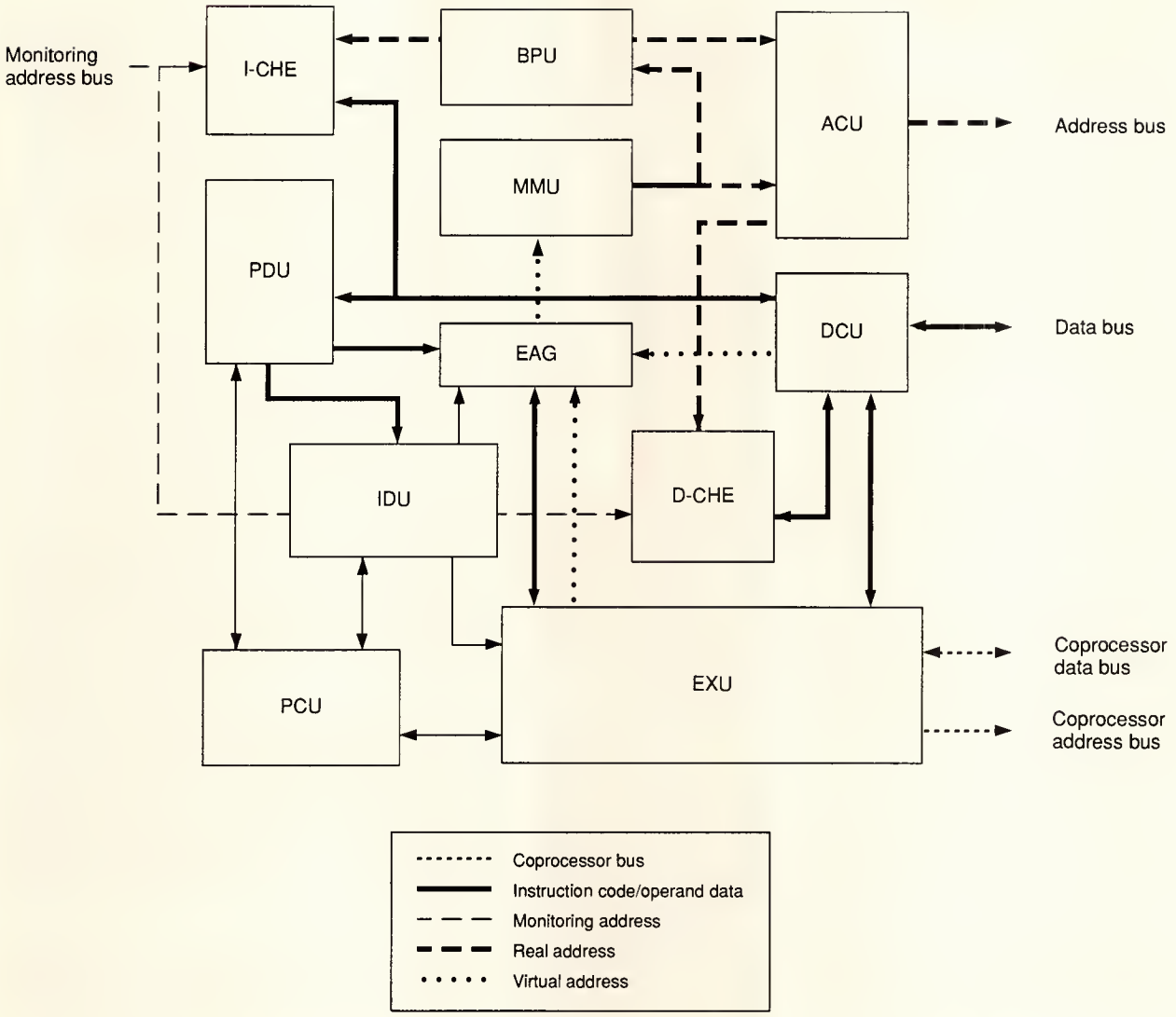


Figure 3. The internal structure of the V80.

Although the pipeline processing technique was actively adopted in the V60 and V70, its basic control had been changed due to a performance analysis conducted on the V70. We considered the depth, length, and distribution of loading at each pipeline stage to exhibit the effectiveness of the other two techniques in the pipeline.

With the new techniques, the V80 can execute about 100 basic instructions such as transfer, integer arithmetic, or branch in two clock cycles, regardless of the operand types and addressing modes. To integrate the hardware to implement these new techniques and to operate at 33 MHz, we developed a new double-metal-layer CMOS process that is based on a 0.8-μm design rule.

Implementation of the V80

We focus now on the pipeline structure and the techniques used to achieve the target performance in the V80. Note that the on-chip memory and the branch prediction mechanism are related to the pipeline processing.

Structure and control. The V80's 11 independent hardware units correspond to each pipeline stage, as shown in Figure 3. The figure also illustrates the flow of addresses and data in a chip. Each of the 11 units processes a basic instruction in two asynchronous clock cycles.

Access control unit. The ACU controls the interface between units on the chip and on-chip cache memories, external main memory, or peripheral I/O devices. It also arbitrates the access requests from other units in a predetermined priority. The ACU controls the detection and avoidance of memory hazards.

Data control unit. The DCU contains four read buffers and two write buffers, which are controlled by the access requests accepted in the ACU. A word alignment function for data read and write operations solves the issue of unaligned data in the address space. The DCU also contains a data processing engine to execute the comparison and search operations for character-string manipulation instructions.

Instruction cache. The I-CHE is a 1-Kbyte cache memory constructed by tag memory and data memory for instruction code. Real addresses from the branch prediction unit (described later) permit access to the cache memory in every clock cycle. The 16-byte-block I-CHE employs a two-way set-associative organization. If a hit occurs, a 4-word instruction code transfers to the predecode unit (described later). When a miss occurs, a block is replaced with the data transferred from the DCU. The I-CHE can purge a block by monitoring the external monitoring address.

Data cache. The 1-Kbyte cache memory D-CHE stores operand data. The basic structure of the D-CHE is the same as that of the I-CHE. It uses a write-through method as write-control strategy. For a read access, a 4-word code, which is associated with the real address of the DCU, transfers to the DCU. For a write access, the D-CHE rewrites the block and requests a write bus cycle to the DCU after the read operation for a block.

Predecode unit. The PDU stores the prefetched instruction code to a 16-byte prefetch queue. The PDU also separates the instruction code into fixed-length information to make the decoding easy for the instruction decode unit. If room exists in the prefetch queue for another instruction, the PDU requests access for the next instructions to the BPU.

Branch prediction unit. The BPU controls instruction prefetch operations and branch prediction processing. For branch prediction, it uses a 64-entry branch prediction table. The BPU registers a pair of branch base and branch target addresses to the table whenever it detects an unpredicted branch instruction.

Instruction decode unit. The IDU decodes opcode and addressing information transferred from the PDU. Results of the opcode decoding specify the operation of the execution unit. Addressing information results specify the effective address calculation in the effective address generator. For a branch instruction, the IDU calculates both branch base and branch target addresses for the EAG.

Pipeline control unit. The PCU detects and solves any pipeline hazards caused by the instruction processing. The PCU treats flag and register hazards, while the ACU treats memory hazards. A register hazard indicates to the IDU that it should stop the decoding process until the result is fixed. Release of a flag hazard resumes the branch prediction process.

Effective address generator. The EAG calculates a 32-bit effective address for the memory operand upon receiving instructions from the IDU. Two carry-save adders and a carry-propagate adder let the EAG add base, index, and displacement values at the same time. The EAG can perform an addition in every clock cycle. The calculated virtual address transfers to the memory management unit.

The EAG also contains and maintains five program counters that correspond to each pipeline stage. The sixth program counter, which corresponds to the prefetch stage, appears in the BPU as the prefetch pointer.

Memory management unit. The MMU translates virtual addresses to real addresses and detects protection violations. A 64-entry, two-way set-associative TLB achieves a 98 percent hit ratio (calculated from trace data). When a miss occurs in the TLB, the MMU itself, without the help of the microprogram in the EXU, replaces a TLB entry. Least recently used (LRU) logic in the MMU determines the replacement candidate.

Execution unit. The EXU executes, interrupts, and processes exceptions. It contains general registers, sixteen 32-bit scratch-pad registers, a 64-bit barrel shifter, a 32-bit binary ALU, a two-digit decimal ALU, and a parallel multiplier implemented by a second-ordered Booth's algorithm. Four of the general-purpose registers can be read simultaneously since they are connected to four separate buses. Two of these registers function as indexes and base values to calculate an effective address in the EAG. The EXU is a microprogram-controlled machine for which 8K steps of microprogram are stored in a microcode ROM. (A step is a 44-bit-wide vertical microinstruction.) The EXU executes the microprogram according to instructions from the IDU.

Parallel operation. Each pipeline unit contains a seven-stage pipeline structure in which the 11 units are distributed. Figure 4 demonstrates the instruction processing flow in the V80. In this figure, effective address generation in the EAG (stage 4) follows instruction decoding in the IDU. Because all other steps process simultaneously, the V80 can process six instructions in parallel.

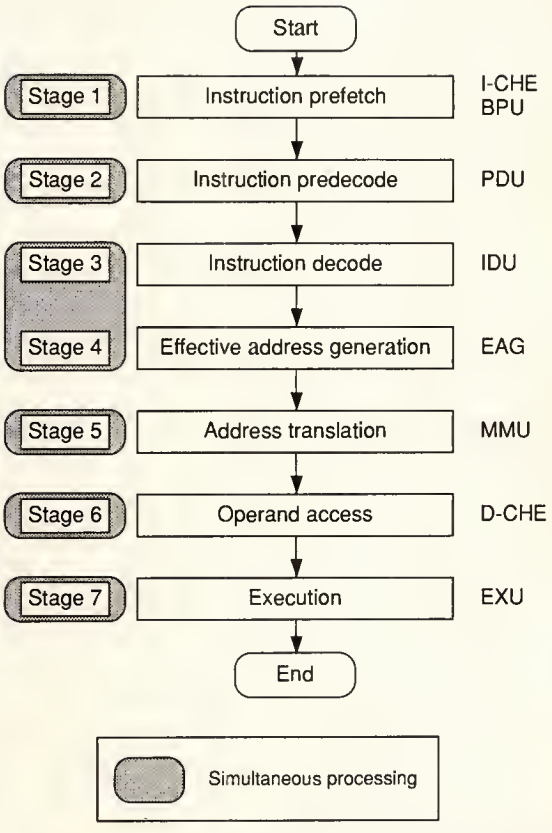


Figure 4. Instruction processing flow.

Figure 5 indicates an example of pipeline processing in the V80 for the instruction stream including:

- a memory operand,
- a complex addressing mode (indexed addressing mode), and
- a branch instruction in case a branch prediction succeeds.

This figure shows that the V80 processes six instructions simultaneously and one instruction every two clock cycles.

Pipeline processing. Ideally, pipeline processing offers the maximum performance when instructions are supplied to each stage continually. Generally however, factors such as the variable-length instruction format, complex instructions, and so on disturb ideal operation in the typical CISC (complex instruction-set computing) MPUs.

To minimize disturbance, we considered the following techniques:

- decreasing instruction loading in each stage,
- preventing stoppage of the pipeline by branches,
- improving exceptional processing, and
- ensuring rapid recovery after interference between instructions.

The third item, exceptional processing, means improved performance for seldom-performed processing

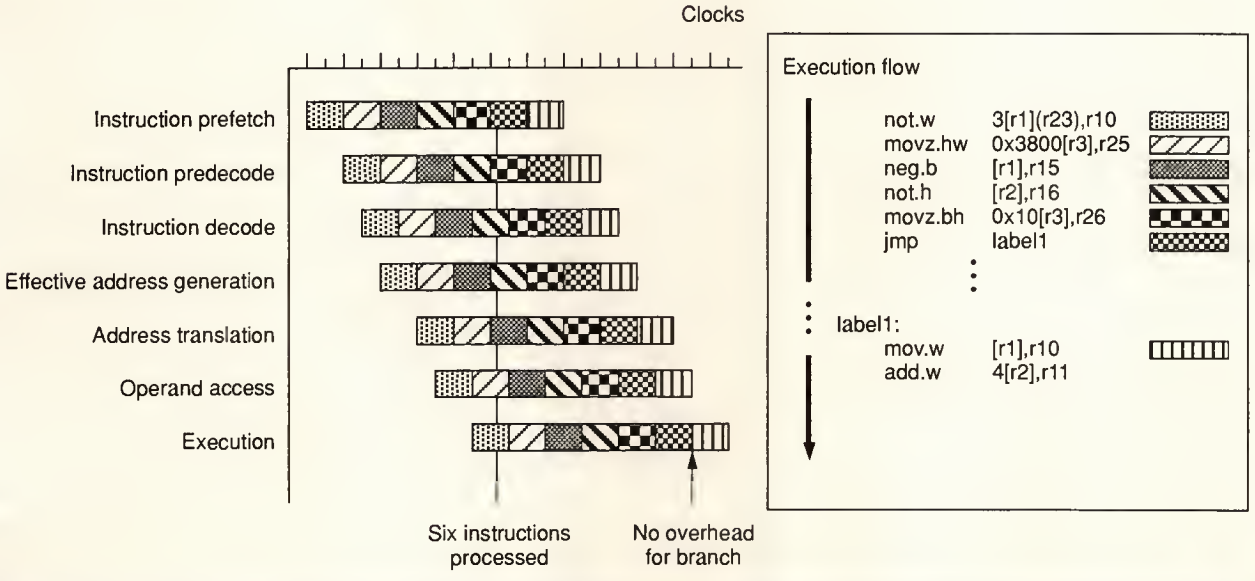


Figure 5. An example of the pipeline stream.

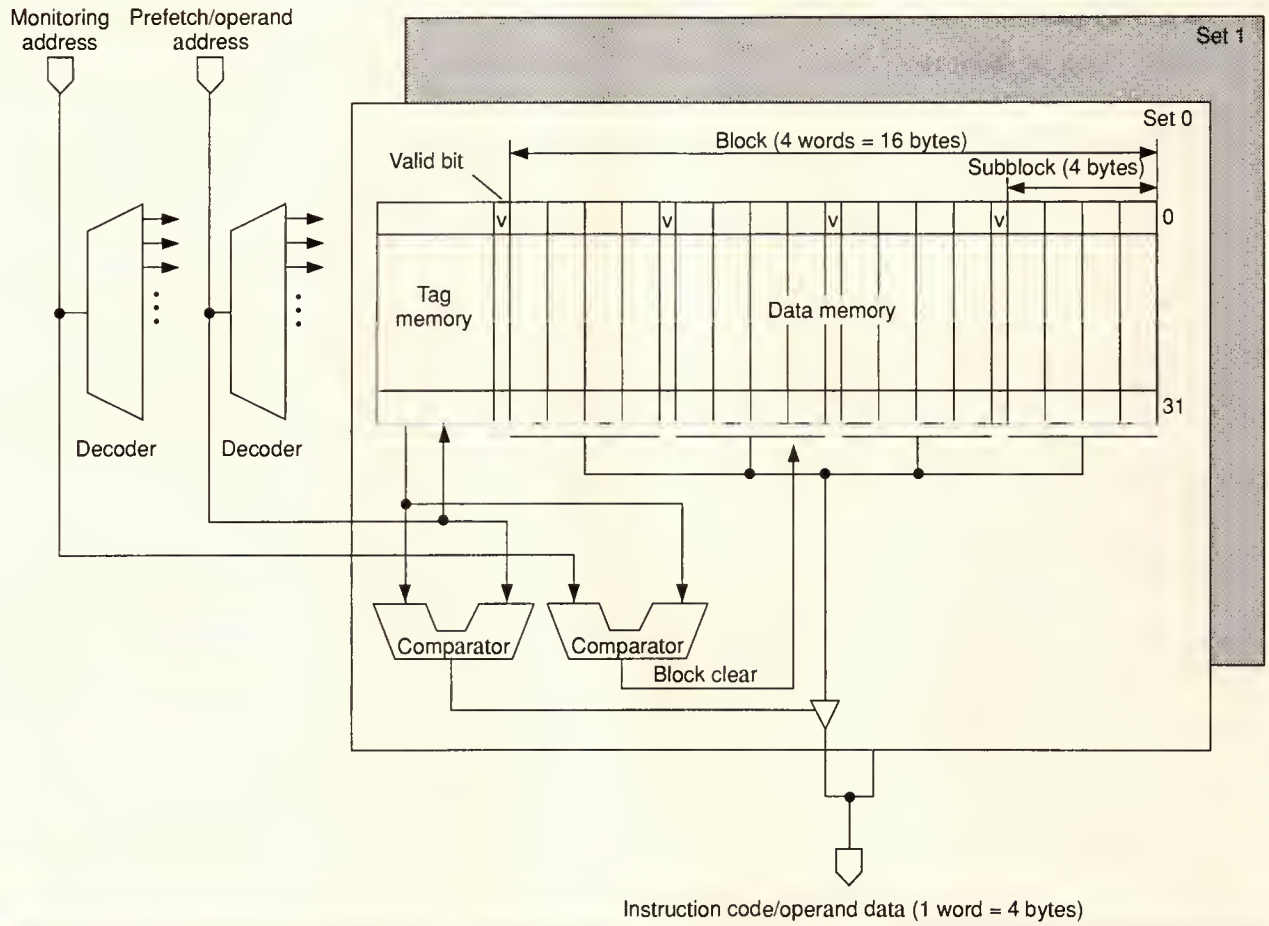


Figure 6. Configuration of the cache memory.

such as a TLB replacement or a floating-point operation. We describe some solutions introduced into the V80 based on these points. To realize these solutions, we applied trace data obtained from an application system using the V70 and a pipeline-simulation model.

On-chip cache memories. Cache memory is essential to continually supply instructions and operands to the pipeline. With only an external cache memory supporting a system, the system cannot duplicate the performance, timing design, and system cost of high-speed MPUs. An on-chip cache simplifies the timing design. A system that includes a two-level cache memory using the on-chip cache as the primary cache can have a moderate capacity.

The V80 contains two independent cache memories for instruction and data to avoid access conflict. Each cache memory has the same configuration: 1-Kbyte capacity, two-way set-associative organization, and a write-through method such as write control for the data cache. From simulations, we obtained a 93 percent hit rate for this model. Figure 6 is the block diagram of each cache memory.

The write-through method makes it easy to keep coherency between on-chip cache and main memory in a multiprocessor system. When a read miss occurs, a burst transfer of 4, 8, or 16 bytes replaces the data; one block (16 bytes) transfers from main memory to the on-chip cache in five clock cycles. When a write miss occurs, a write-allocate function creates a new block, as a read access can be expected just after the write access.

Branch prediction mechanism. Performance tends to bottleneck in pipeline-processed MPUs when a branch instruction purges the pipeline stream. As shown in the trace data, typical application programs for the V70 executed 15 to 30 percent of the branch instructions. The V80 employs a branch prediction mechanism to keep the disturbance to a minimum.

The BPU, related to the instruction prefetch and predecode stages, processes branch predictions. The PCU also helps the prediction operation. Figure 7 shows the configuration and concept of the mechanism. The BPU's prefetch pointer (PFP) accesses the instruction cache and controls the instruction stream to be prefetched. The BPU branch prediction table (BPT) can register 64 pairs of branch base addresses and branch target addresses. When an unpredicted branch instruction executes, the BPT registers the base address and target address.

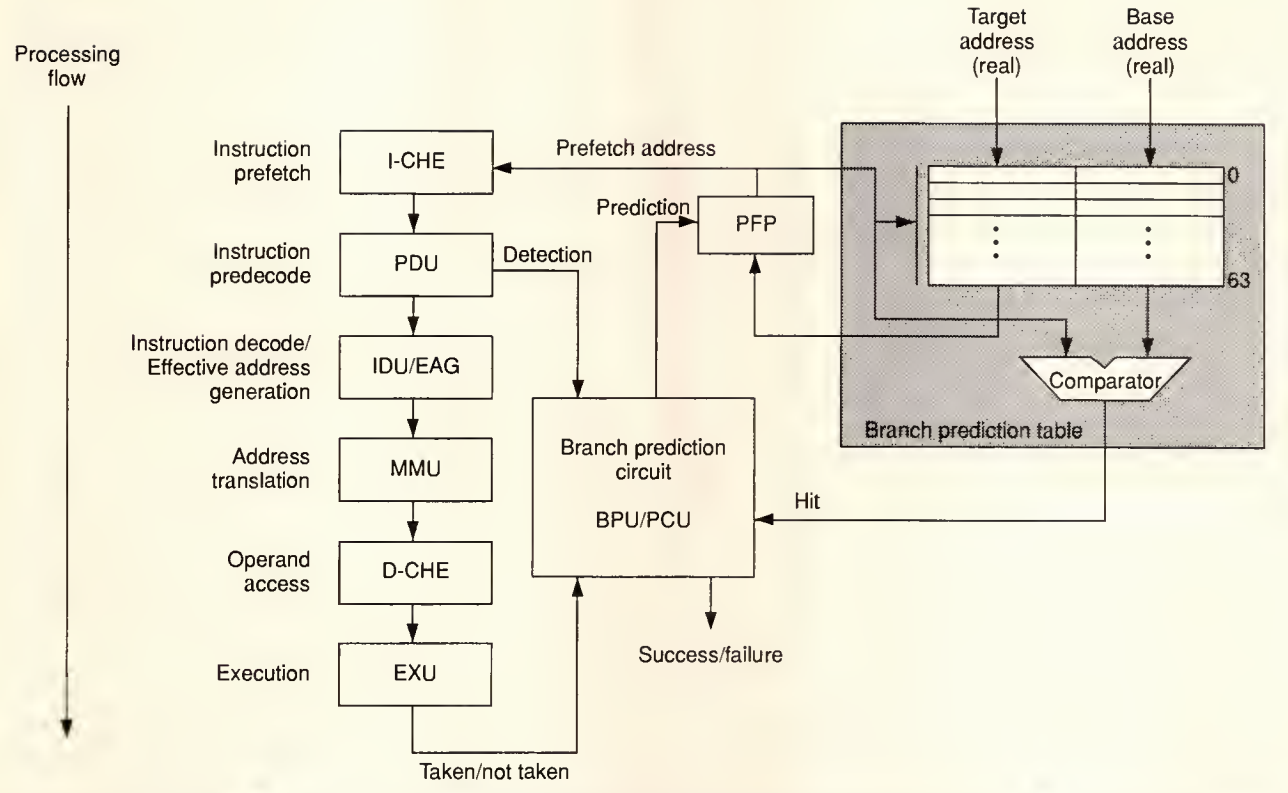


Figure 7. A block diagram of the branch prediction mechanism.

If a prefetch address pointed to by the prefetch pointer hits in the BPT, the prefetch pointer is set to the branch target address registered in the BPT. To perform this control, the BPT uses 1) valid bits, offset in a word for the top address of target instruction, and 2) the byte length of the branch instruction as auxiliary information.

A predicted instruction stream moves through the predecoding, decoding, EAG, address translation, and operand access stages. An instruction following a conditional branch executes without delay if the branch is taken and the prediction is correct. The V80 can process a branch instruction in the same way as other basic instructions using the prediction mechanism. Figure 8 on the next page illustrates the V80's effectiveness. When a prediction fails, a branch operation loses five clock cycles by sequential processing, and simultaneous registration to the BPT occurs.

The V80 performs the prediction operation for all static (PC-relative) branch instructions including loop instructions. Simulation produced a 70 percent hit ratio for the branch prediction. In the simulation, branch instructions executed at a 20 percent ratio in the total instructions.

Two-stage decoding. Generally, in an MPU employing a variable-length instruction format, instruction decoding is most critical for performance. The V80 uses various types of instruction formats as a typical CISC MPU. Especially at high-speed clock rates, the delay time of decoding circuits causes serious problems. The V80 processes the instruction decoding by a two-level decoding procedure that involves the predecode and decode stages. Figure 9 shows the mechanism of two-level decoding. In this figure, the I-CHE supplies the instruction code via the instruction bus. The PDU detects the top address of an instruction and separates it into pairs of fixed-length information such as instruction code, addressing modes, register number, and displacement by the pre-predecoder and predecoder. The IDU decodes a fixed-length instruction generated by the PDU and generates the control information for the EAG and EXU.

The PDU has a 16-byte (4-word) prefetch queue and a predecode queue for two instructions that function as a pipeline buffer. Also, it contains a branch target buffer, which prefetches a target instruction independently of prediction to reduce overhead at the taken branch. With two-stage decoding, the V80 can decode a long, complex instruction in two clock cycles at a 33-MHz clock speed.

Coprocessor bus. The instruction set of the V60/V70/V80 can be extended by using the μPD72691 floating-point coprocessor.[6,7] The V80 includes four basic operations for 32/64-bit floating-point data. However, the coprocessor adds the following features to the floating-point arithmetic instructions of the V80:

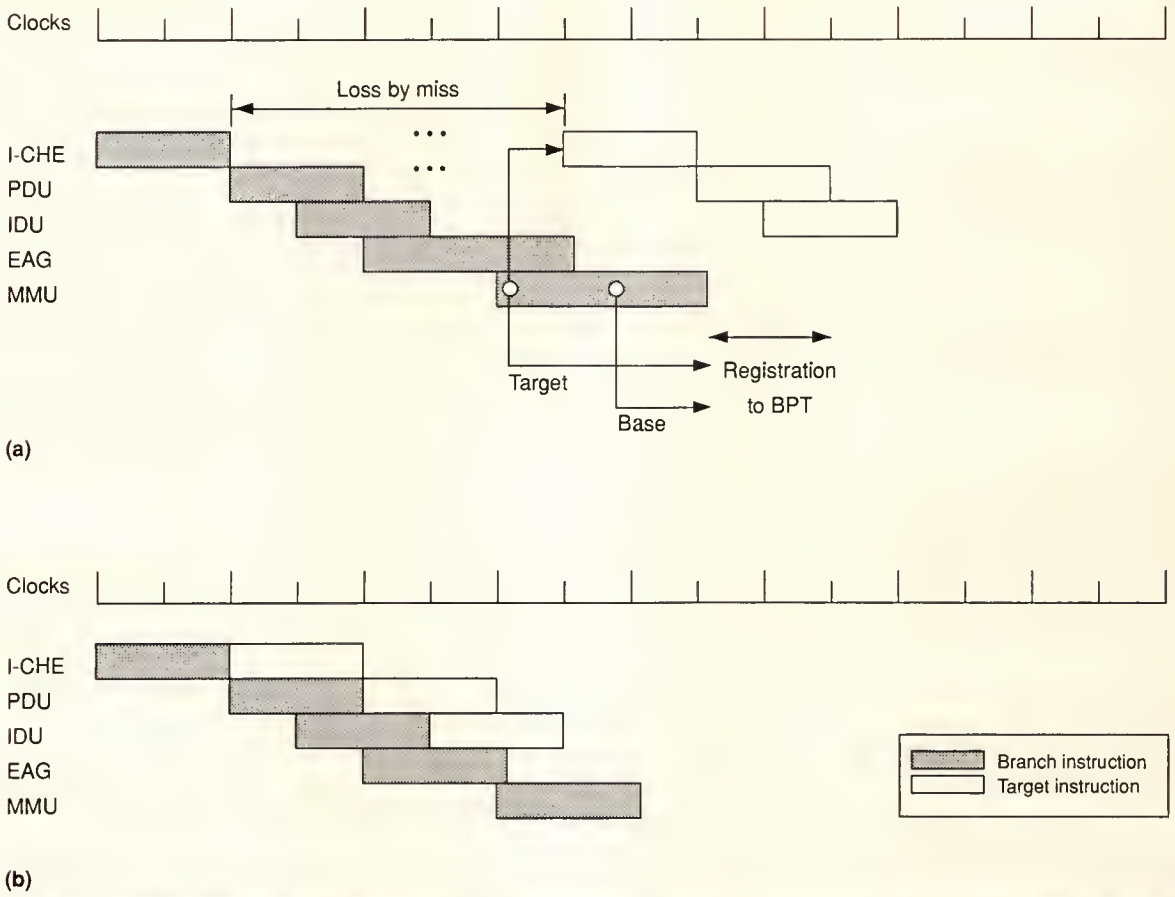


Figure 8. Effects of the branch prediction mechanism: prediction miss (a) and hit (b).

• trigonometric, inverse trigonometric, exponential, and logarithmic functions;
• vector (3/4 elements)/matrix (3 × 3/4 × 4 elements) operations; and
• 80-bit floating-point data.

The V60 and V70 connect the coprocessor on the system bus and communicate with it in a predetermined sequence. To reduce the time required for the protocol that is performed between the V80 and the coprocessor, the V80 uses dedicated data and address terminals called the coprocessor bus. The protocol, which needs bus accesses for transmitting data and commands and receiving results and status, may conflict with the MPU pipeline stream. Positioned as an extension of the execution stage, the coprocessor bus can disturb the instruction prefetch and operand-read operations during the protocol process. Up to 32 coprocessors can be connected to the coprocessor bus.

The protocol runs three times faster than it does in the V70 at the same clock rate. No overhead is needed to control the coprocessor with the parallel operation of the coprocessor.

Improving address translation. To reduce misses, we enlarged the capacity of the TLB in the V80 to 64 entries from the 16 entries in the V70. A microprogram in the V70's execution stage replaces the entry, causing execution to be disturbed. The V80 MMU replaces a TLB entry with a hardware sequencer when a miss occurs and executes the data simultaneously.

Also, the V80 contains a page table base (PTB) buffer that holds the contents of the LRU address translation table in the main memory. A virtual address translates to a real address after referencing a register set (area table base/length register) and a two-level table in main memory (area table/page table). The buffer caches the first table (ATE, or area table entry) because the next miss may occur using this table by reference locality.

In the best case, entries can be replaced in six clock cycles with this improvement.

Other features. The V80 also contains some special hardware such as engines for complex instructions. EXU microprograms in the V70 process these instructions. One of the solutions for avoiding performance drops by the biased loading is to distribute the loading to other stages.

An example is the string-manipulation engine that

READER-SERVICE CARD

IEEE MICRO

April 1990 issue (card void after October 1990)

Name _____

Title _____

Company _____

Address _____

City _____ State_____ ZIP _____

Country _____ Phone (_____) _____

Please send (Circle those you want)

201 Publications catalog
202 Membership information
203 Student membership information
204 Senior membership information
205 *IEEE Micro* subscription information

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150 151 152 177 178 179
153 154 155 180 181 182
156 157 158 183 184 185

159 160 161 186 187 188
162 163 164 189 190 191
165 166 167 206 207 208

168 169 170 209 210 211
171 172 173 212 213 214
174 175 176 215 216 217

Product Information 1
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	—
11	31	51	71	91	111	131	—
12	32	52	72	92	112	132	192
13	33	53	73	93	113	133	193
14	34	54	74	94	114	134	194
15	35	55	75	95	115	135	195
16	36	56	76	96	116	136	196
17	37	57	77	97	117	137	197
18	38	58	78	98	118	138	198
19	39	59	79	99	119	139	199
20	40	60	80	100	120	140	200

READER-SERVICE CARD

IEEE MICRO

April 1990 issue (card void after October 1990)

Name _____

Title _____

Company _____

Address _____

City _____ State_____ ZIP _____

Country _____ Phone (_____) _____

Please send (Circle those you want)

201 Publications catalog
202 Membership information
203 Student membership information
204 Senior membership information
205 *IEEE Micro* subscription information

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150 151 152 177 178 179
153 154 155 180 181 182
156 157 158 183 184 185

159 160 161 186 187 188
162 163 164 189 190 191
165 166 167 206 207 208

168 169 170 209 210 211
171 172 173 212 213 214
174 175 176 215 216 217

Product Information 2
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	—
11	31	51	71	91	111	131	—
12	32	52	72	92	112	132	192
13	33	53	73	93	113	133	193
14	34	54	74	94	114	134	194
15	35	55	75	95	115	135	195
16	36	56	76	96	116	136	196
17	37	57	77	97	117	137	197
18	38	58	78	98	118	138	198
19	39	59	79	99	119	139	199
20	40	60	80	100	120	140	200

SUBSCRIPTION CARD

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

❑**YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only **$19** for a year's subscription (six issues).

Society: _____ IEEE membership no.: _____

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate $9.50 for three bimonthly issues.

❑**YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE (but not a member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other qualified professional technical society, pay the sister-society rate of only **$35** for a year's subscription (six issues).

Organization: _____ Membership no.: _____

Full Signature *Date*

Name

Street

City

State/Country *ZIP/Postal Code*

❑ Payment enclosed

❑ Charge to ❑ Visa ❑ MasterCard ❑ American Express

Charge-card number

Expiration date

Month *Year*

Prices valid through 12/31/90
4/90 Micro

Change orders also taken by phone:
(714) 821-8380 8a.m. to 5 p.m. Pacific time
Circulation Dept.
PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

_____

_____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

Reviewers needed. *If interested, send professional data to Joe Hootman, EE Dept., University of North Dakota, PO Box 7165, Grand Forks, ND 58202.*

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

IEEE Micro
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

Editorial comments

I liked: _____

_____

_____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

For reader-service inquiries, see other side.

Reviewers needed. *If interested, send professional data to Joe Hootman, EE Dept., University of North Dakota, PO Box 7165, Grand Forks, ND 58202.*

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

IEEE Micro
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY
Circulation Dept.
PO Box 3014
Los Alamitos, CA 90720-9804
USA

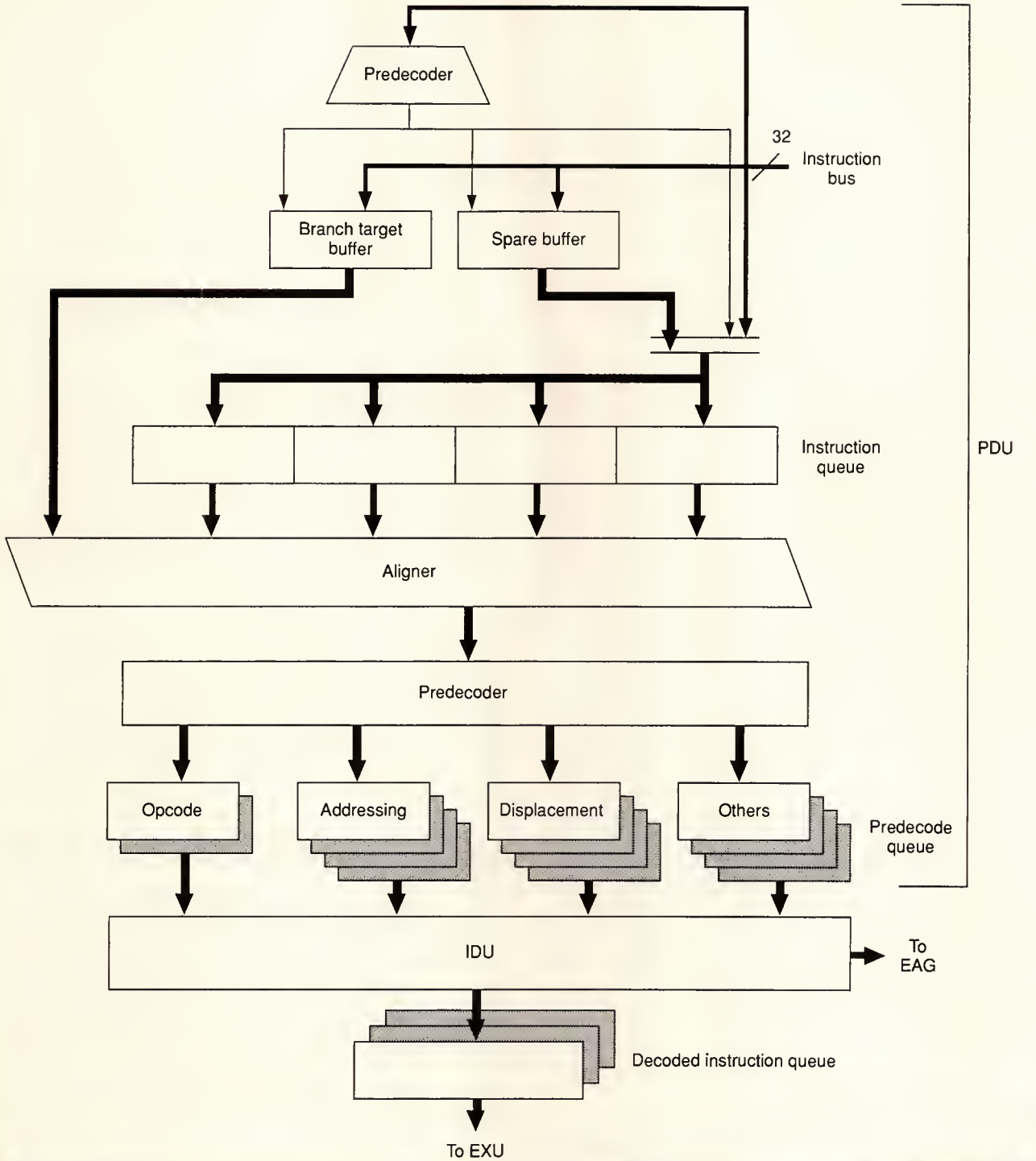


Figure 9. Configuration of two-stage decoding.

supports the execution of string-manipulation instructions in the DCU. The engine can perform data processing such as calculation of the string length and comparison of characters. Also, it controls a buffer for character data to reduce bus accesses. With the buffer, a 4-byte character transfer operation drives a 1-word bus cycle. The V80 performs string-manipulation instructions five times faster than the V70 at the same clock rate. The performance is limited mainly by the maximum bus transfer capacity.

Another solution is to adopt dedicated hardware for special instructions instead of a microprogram. The V80 includes basic floating-point arithmetic instructions that are executed on chip. Floating-point opera-

Table 2.
V70 and V80 execution clocks.

Function	Instruction/ condition	V70	V80
Transfer	MOV.W reg, mem	4	2*
	MOV.W mem, reg	4	2*
Integer operation	ADD.W reg, mem	2	2
	ADD.W mem, reg	4	2*
	ADD.W mem, mem	8	4*
Integer multiply	MUL.W	23	9*
Integer divide	DIV.W	43	39
Shift	SHA.W	17	3*
Branch	Taken	11	2*
	Not taken	4	4
Procedure call	CALL + RET	44	21*
Bit field	EXTBFZ	30	10*
	INSBFZ	28	10*
Character string	MOVCU.B (n bytes)	30 + 5n	19 + 1.25n*
Floating-point operation (32 bits)	ADDF.S	120	36*
	MULF.S	116	44*
	DIVF.S	137	75
Floating-point operation (64 bits)	ADDF.L	178	75*
	MULF.L	270	110*
	DIVF.L	590	553
Return from interrupt	RETIS	80	22*
Context switching (maximum context)	LDTASK (44 words)	347	157*
	STTASK (44 words)	200	121*
TLB replacement	Same area	58	11*
	Another area	58	6*
Interrupt response	Until the start of handler	165	27*

*Less than one-half cycle

Table 3.
V70 and V80 performance comparison.

Benchmark	Purpose	V70 (20 MHz)	V80 (33 MHz)
Dhrystone (version 1.1)	Integer operation	1	4.2 (relative)
EDN-E	Character-string manipulation	60 µs	9 µs
Bit-string transfer	Bit-string manipulation	137 µs	7 µs
Task switching	Real-time operation	36 µs	11 µs

tions based on the ANSI/IEEE 754 standard[8] require that exceptional operations such as those using infinity, not-a-number, or abnormal numbers be checked and detected. These complex specifications require many internal operations and judgments to perform an entire operation. The V70 overhead includes the operation for actual calculations, since it was done purely by microprogram. To reduce this overhead, the V80 uses a mechanism that checks and detects the exceptional operations. The V80 also has a mechanism that separates and combines mantissa and exponent parts of floating-point data. With these mechanisms, the V80 solves floating-point operations three times faster than the V70 at the same clock rate. For floating-point multiplication, a multiplier that can perform a 32-bit integer multiplication in four clock cycles achieves almost the same performance as for floating-point additions and subtractions. The multiplier uses the second-ordered Booth's algorithm, and it employs four carry-save adders and a carry propagate adder to achieve 8-bit processing at every clock cycle.

In addition, the V80 distributes hardware functions, which were processed by microprograms in the V70, to achieve task switching and interrupt/exception handling.

Performance comparison. Table 2 lists the V80 execution clock numbers for some instructions and processing. It proves that the V80 achieves two to four times the performance for the V70 at the same clock rate. Table 3 also compares the performance between the V70 and V80 by using the following benchmarks for applications:

- Dhrystone (version 1.1);
- *EDN*-E (searches for a 15-character pattern in 120 characters);
- bit-string transfer (transfers 1,000-bit data from/to memory with logical operations such as And); and
- task switching (searches for the highest task and switches the following resources: two privileged registers called TKCW and SBR; 13 general registers; three stack pointers; and three pairs of virtual memory management registers).

The table shows the well-balanced performance improvement of the V80.

System support functions

The V80 makes use of three types of bus cycles. The primary bus cycle, the same as in the V70, takes two clock cycles and is issued in normal address mode. The operation clock speed of the V80 is higher than 25 MHz. Designing a no-wait memory system with the normal address mode while considering a maximum delay time for the terminals such as address bus and status information is difficult and expensive. Generally, some wait states might be required for a bus cycle.

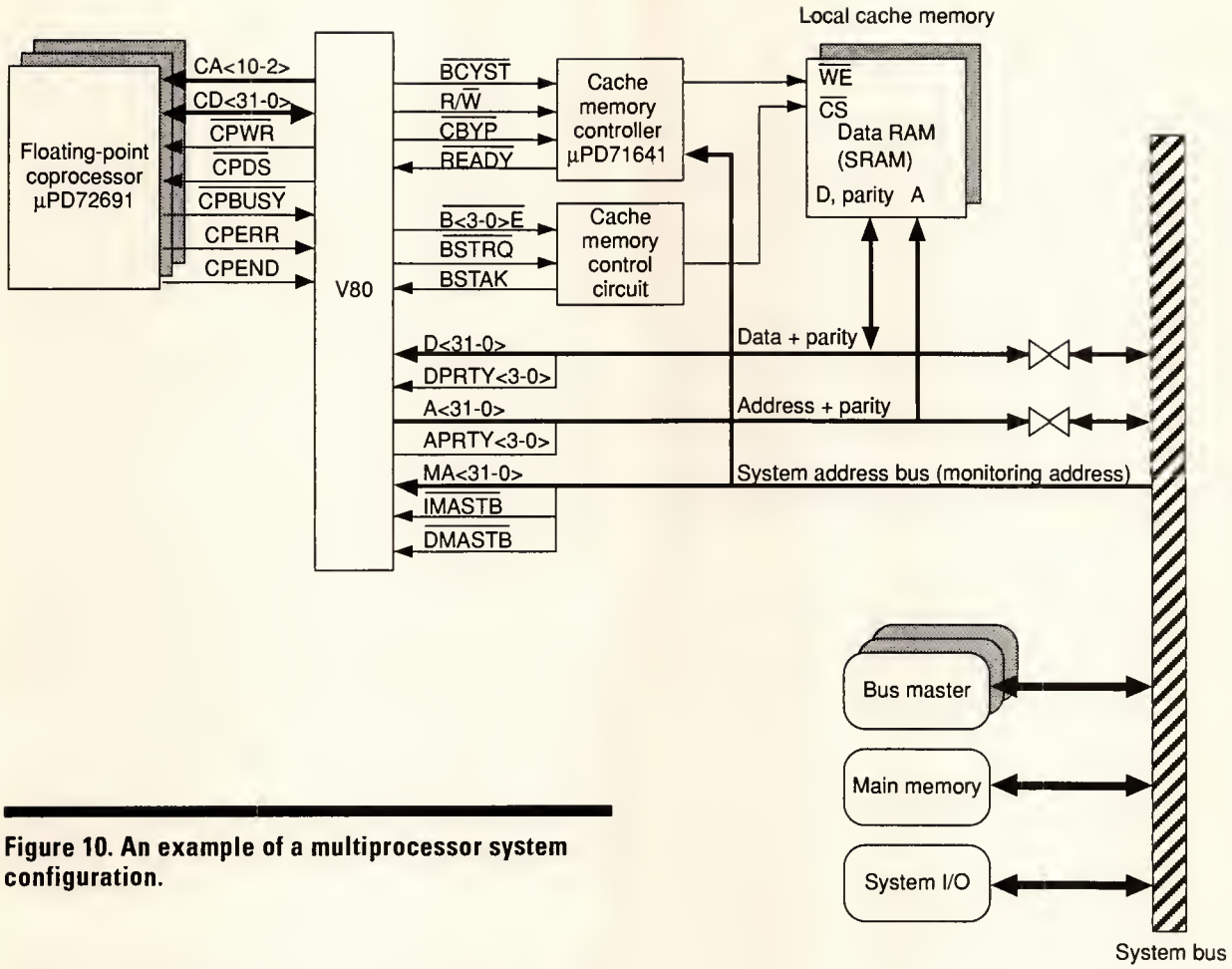


Figure 10. An example of a multiprocessor system configuration.

One of the solutions, to use popular DRAMs with minimum overhead, adopts an on-chip cache memory. As already mentioned, the on-chip cache memory of the V80 decreases external bus traffic.

The advanced address mode offers another solution, and it is the second type of bus cycle in the V80. In this mode, output of addresses and transfers of data behave just like they do in pipeline processing. When transferring data, the V80 outputs address and status signals before one clock cycle transpires. By using the DRAMs with page mode, a one-clock advantage can be achieved. The V80 operating at 25 MHz guarantees an access time of 80 nanoseconds.

The advanced bus mode and the normal bus mode can be switched dynamically for every bus cycle by a control terminal.

On-chip cache memory support. The third type of bus cycle in the V80, the burst mode, is applied in the replacement of on-chip cache memory. When a miss occurs in the on-chip cache memory, the V80 drives continual read bus cycles for a replacement. A replacement takes one, two, or four bus cycles, according to the setting of the CHCW. This means that a miss is detected by every one, two, or four words in a block of the cache.

Four subblocks respond to four contiguous words and make up a block. The V80 replaces one, two, or four subblocks, including the subblock that causes a selective miss. Increasing the number of replacements leads to improving the hit ratio of the on-chip cache memory. On the other hand, it may cause additional overhead at every miss. Consequentially, a number of replacements can be selected, according to memory system and program behavior.

Another improvement to reduce replacement time is the burst mode for a bus cycle, as mentioned earlier. When the CHCW selects four bus cycles for a 4-word replacement, the first bus cycle takes two clock cycles. After the V80 checks to see whether an external system can accept the burst-mode bus cycle, the remaining three bus cycles read data at each of the next three clock cycles. In burst mode a 4-word data transfers in five clock cycles, while it takes eight clock cycles to transfer without this mode. The burst mode can combine with the advanced address mode and be applicable for the external cache memory and DRAMs using nibble or page mode.

Multiprocessor support. Figure 10 shows a multiprocessor configuration. To construct a multiprocessor system, some potential problems must be solved.

Table 4.
Interprocessor synchronization instructions.

Instruction	Function
CAXI	Compare and exchange with interlock
TASI	Test and set with interlock
ADDI	Add with interlock*
SUBI	SUB with interlock*
ANDI	And with interlock*
ORI	Or with interlock*

*New instructions added to the V80

The first one is to reduce system bus traffic. System bus traffic influences system performance because multiple MPUs are connected to the system bus. Using the coprocessor bus, as explained earlier, reduces traffic between the floating-point coprocessor and the V80. Each V80 may need a local cache to avoid excess bus traffic.

The next problem is to keep coherency between the on-chip cache and main memory. It is complex to keep on-chip cache coherency because one MPU can change the main memory in a multiprocessor environment. To ensure coherency, older MPUs purge the on-chip cache at every memory access by another bus master. The V80 purges just one block, including the data that is rewritten by another bus master, by monitoring the system bus. In Figure 10, this action is indicated by the input from the system address bus to the monitoring terminals of the V80. The V80 experiences no timing restriction and no loss time when monitored by dedicated terminals. Conventional MPUs use a common address bus for the monitoring.

Software synchronization proceeds according to the instructions shown in Table 4 on the next page. Those instructions interlock individual accesses for memory operands by asserting the bus lock and cache bypass signals. We added four instructions to the V80 to improve the performance of interprocessor synchronization. These instructions can be applied to access semaphores and modify common data and pointers.

High-reliability support. The V80 uses three functions to support high-reliability systems:

- functional redundancy monitoring (FRM),
- bus error exception, and
- parity generation and error detection.

The first two functions have already been adopted in the V60 and V70. By using those functions, the MPU module in the V80 can offer higher reliability. However, to construct a high-reliability system, the MPU as well as an external memory system should be highly reliable. Generally, some types of error code such as parity or CRC (cyclic redundancy code) are added to the main memory to detect an error that might occur in the main memory or between the main memory and the MPU. It is difficult to generate and detect an error code by a suitable external circuit for high-speed bus cycles with a 33-MHz operation clock. As a wait state would be needed for every bus cycle to adopt an error code, the system would experience a performance drop.

To avoid that overhead, the V80 contains internal parity generation and error detection circuits. The parity signals are added to every 8-bit address and data bus, four address parity signals for the address bus and four data parity signals for the data bus. Each terminal generates an even parity signal for each 8 bits in the write bus cycle. In the read bus cycle, the V80 checks the parity signals transferred from the main memory, while the address parity terminal generates the parity signal. If a mismatch occurs between the state of each data parity terminal and internally calculated parity signal according to the value of the data bus, the V80 causes a parity error exception. It treats the exception in the same manner as a bus error exception.

This function simplifies a highly reliable memory system, because the MPU needs no external circuits to generate and check parity.

We've described our implementation of the 32-bit V80 microprocessor, focusing on pipeline processing and introducing the system support functions. We used cache memories and a branch prediction mechanism to improve pipeline processing. Various hardware facilities replaced the usual microprogram to ensure high performance.

The V80 integrates 980,000 transistors on a 14.49-mm × 15.47-mm die. ROM, RAMs, and PLAs make up 63 percent of these transistors. The biggest circuit is the microcode ROM, which shares 360,000 transistors. On the other hand, various system support functions of the V80 made the construction of the hardware system easy.

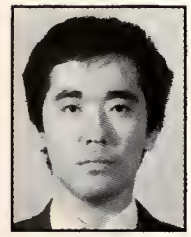
The V80 is suitable for multiprocessor and highly reliable system configurations.

Acknowledgments

We thank all of the members of the V80 project. The limited space does not allow us to acknowledge each of them here, but we would like to give special thanks to K. Kani and I. Fujitaka for their support. We also thank Y. Yano and S. Burns for their helpful suggestions.

References

1. Y. Yano et al., "A 32-Bit Microprocessor with On-Chip Virtual Memory Management," *ISSCC Digest of Technical Papers*, Feb. 1986, pp. 36-37.
2. H. Kaneko et al., "A 32-Bit CMOS Microprocessor with Six-Stage Pipeline Structure," *Proc. Fall Joint Computer Conf.*, Nov. 1986, pp. 1000-1007.
3. H. Monden, "Introduction to ITRON, the Industry-Oriented Operating System," *IEEE Micro*, Apr. 1987, pp. 45-52.
4. Y. Yano et al., "V60/V70 Microprocessor and its Systems Support Functions," *Compcon 88 Digest of Technical Papers*, Mar. 1988, pp. 36-42.
5. S. Kimura et al., "Implementation of the V60/V70 and Its FRM Function," *IEEE Micro*, Apr. 1988, pp. 22-36.
6. T. Nakayama et al., "An 80b, 6.7MFLOPS Floating-Point Processor with Vector/Matrix Instructions," *ISSCC Digest of Technical Papers*, Feb. 1989, pp. 52-53.
7. T. Nakayama et al., "A 6.7MFLOPS Floating-Point Coprocessor with Vector/Matrix Instructions," *IEEE J. of Solid-State Circuits*, Vol. 24, No. 5, Oct. 1989, pp. 52-53.
8. *ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic*, IEEE Computer Society Press, Los Alamitos, Calif., 1985.



Hiroaki Kaneko is a project member of NEC Corporation's V80 development team. He has developed 16-bit and 32-bit microprocessors in NEC's V series since 1982. Kaneko received his BE and ME degrees in electrical engineering from the Tokyo Denki University. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan and the Information Processing Society of Japan.

Nariko Suzuki is one of the system-level designers of the V80. She has worked to develop the 32-bit V series since 1984. Suzuki received her BE degree in information science from the Tokyo Agriculture and Technology University.

Hiroshi Wabuka, a project member of the V80 design team, has worked with the circuit design of the V series since 1982. Wabuka received his BE and ME degrees in electrical and electronic engineering from the Himeji Institute of Technology. He is a member of the IEICE.

Koji Maemura is also a system-level designer of the V80. He has been engaged in the development of the 32-bit V series since 1984. Maemura joined NEC IC Microcomputer Systems Ltd. after receiving his BA degree in mathematics from Yokohama University.

Questions concerning this article can be directed to Hiroaki Kaneko, Advanced Products Dept., Microcomputer Div., NEC Corporation, 1753, Shimo-numabe, Nakahara-ku, Kawasaki, Kanagawa 211, Japan.

Microprocessor Memory Management Units

This article provides an overview of the rationale, principles, and hardware support for virtual memory. It then reviews and compares the design and features of high-end microprocessor MMUs across a common set of criteria. Special attention is paid to Unix requirements and multiprocessor, multiple MMU considerations.

Milan Milenkovic

IBM Corporation

This tutorial describes the current crop of commercial memory management units (MMUs) for 32-bit microprocessors. The discussion includes both complex and reduced instruction-set computers (CISCs and RISCs). The first section reviews the rationale, principles, and issues related to hardware support for memory management and virtual memory. The next section contains brief overviews of the following MMUs: Intel's 80386, i486, and i860; Motorola's 68851 (MMU for the 68020), 68030, 68040, and 88200 (MMU for the 88000 series); the Fujitsu MB86920 (Sparc MMU); and the MIPS R2000/R3000.

Virtual memory management

Historically, the development of multitasking and multiuser systems introduced the need for memory management. Multitasking is essentially the multiplexing of CPU cycles among the tasks resident in main memory to achieve a high overall resource utilization. In such systems, memory management must enforce isolation of distinct address spaces to preserve the system's integrity. Protection fences established by the memory management hardware prevent the executing task from trespassing the boundaries of other tasks. At the same time, the memory manager must also allow controlled sharing of memory to support intertask cooperation and synchronization. Sharing of code, such as in libraries and common utilities, can result in a significant conservation of memory. In addition to meeting these requirements, the memory manager—like any other resource manager—should ensure efficient utilization of memory.

Virtual memory is a memory management scheme distinguished by its capability to execute programs that exceed the size of available physical memory. The address-space size in a virtual memory system is limited only by the number of address bits that the processor can generate (possibly

0272-1732/90/0400-0070$01.00 © 1990 IEEE

augmented by memory management hardware). The capacity of the available physical memory does not affect the address-space size. Virtual memory operates by executing programs only partially resident in memory while relying on hardware and the operating system to bring the missing items into main memory when needed. Virtual memory systems are adaptive in the sense that the system dynamically allows an executing program to expand and fill the available space. Thus, programs generally run faster in hardware configurations that have larger capacities of real (physical) memory.

In virtual memory systems, programmers are not restricted by memory-size considerations, and the programs are portable between hardware environments with different memory capacities. Moreover, support for large virtual address spaces allows for seamless incorporation of the file system and/or remote objects into the address space of a task. The resulting uniformity and linearity of data addressing simplifies programming and increases portability by reducing system and hardware dependencies. Some implementations of virtual memory map the entire operating system into the (virtual) address space of each task. This approach results in protection of finer gradation and in potentially increased efficiency because I/O and system calls do not have to cross the traditional user/system boundaries. Another advantage of virtual memory is that objects automatically migrate between the levels of memory hierarchy on the basis of their frequency of reference. With virtual memory, the operating system has the added flexibility to control performance by varying the amount of real memory allocated to individual tasks.

One can build virtual memory on top of segmentation, paging, or a combination of the two schemes—such as segmentation with paging. In any case, efficient implementation of virtual memory requires significant hardware assistance above and beyond that of the basic underlying memory management scheme.

Rationale and principles of address translation. The clear separation of virtual addresses from physical addresses constitutes a key aspect of advanced forms of memory management. The system forms virtual addresses at program translation and/or link time. These addresses uniquely identify items that belong to one logical address space (say, of a given task). However, the system defers the actual memory allocation—or the determination of the physical-memory load addresses—until runtime. In this way, the operating system manages physical memory the best it can by loading portions of the incoming address spaces into whatever physical memory is available. This freedom is essential to efficient memory management in multitasking and distributed systems in which tasks migrate in and out of main memory in response to state and load variations.

Memory management hardware maps virtual addresses to physical addresses at runtime. The address-translation hardware and the operating system cooperate in this process by using mapping (or page-map) tables. (Note that the terms *mapping tables, page-map tables,* and *page tables* are interchangeable except for instances when segmentation and hierarchy are used.) At runtime, the hardware uses a portion of a virtual address as an index into the table of mapping descriptors. It then uses mapping information contained in the fetched descriptor to produce the physical address of the target item in memory. The operating system places the mapping information into descriptors after loading the corresponding items in physical memory (see the accompanying box).

The key to the versatility of advanced memory

Memory Management, Address Translation, and Virtual Memory

Virtual memory is a memory management scheme that allows execution of programs only partially resident in main memory. It thus allows execution of programs whose address space exceeds the size of the main memory available in a given system configuration. In virtual memory systems, program sizes are limited only by the addressing capability of the processor and not by the size of main memory available on any given machine. Thus, any 32-bit system that supports virtual memory should allow program sizes of up to 4 Gbytes (2^{32}), regardless of the amount of its installed physical memory. Implementation of virtual memory requires special architectural provisions, such as instruction restartability, and ample support from the memory management hardware.

Since the operating system allocates memory dynamically at runtime, programs in systems with virtual memory cannot be bound to specific addresses at compile or link times. Instead, the systems prepare and store programs in executable files with address identifiers relative to the beginning of each individual program. These addresses are sometimes referred to as virtual. Prior to execution, the system loads portions of the program into available sections of physical memory whose addresses generally bear no relationship to the

virtual program-address space. The memory management hardware performs virtual-to-physical address translation (binding) at runtime. The address mapping tables prepared by the operating system in the course of program loading aid the translation process. At runtime, the memory management hardware translates addresses and enforces protection by trapping illegal references. In virtual memory systems, the memory management hardware must also detect missing items that are not present in main memory.

Figure A illustrates the address-translation process in a virtual memory system. This example assumes a 32-bit processor in a system with 32 Mbytes of physical memory. The underlying memory management uses paging with a 4-Kbyte page size. Paging systems allocate memory in fixed-size chunks called pages. In such systems, the MMU conceptually divides physical memory into page frames into which individual pages—obtained by mechanically splitting the program's virtual address space—are stored.

Figure A shows the first six (0-5) pages in the virtual address space of a program. Four of them—P0, P1, P3, and P4—reside in physical (main) memory. All program pages, including P2 and P5, reside on disk. Address translation occurs with the aid of the page-map table (PMT), which has one entry for every virtual page of the related program. Each entry for a page that is present in main memory contains the physical address of the page frame into which the corresponding page is placed by the operating system. For example, the physical address of P4 in main memory starts at 01FFA000 (all addresses are hexadecimal). Given a page size of 4 Kbytes (1000 H), P4 has a corresponding page frame number of 01FFA, which is entered into the PMT. Each PMT entry contains a presence bit to indicate the presence or absence of the related page in main memory. For example, since P2 and P5 are not present in main memory their presence bits indicate that fact by being set to N (not present).

Figure A also illustrates the address-translation process for the data-reference memory access made by the processor on behalf of the example move instruction. The processor operates with and emits the virtual addresses provided in the code of the executing program. In this example, the processor emits address 00004238 to identify the target datum in the program's virtual address space. (R is the register destination.) The address-translation hardware operating in the paging environment splits the virtual address into the page number 00004 and the offset 238. Since pages and page frames have identical sizes, offsets are the same within both and do not need translation. The page number 4 indexes the PMT to obtain the corresponding physical frame number, 01FFA in this case. This value is concatenated with the page offset to produce the physical memory address 01FFA238 in which the target item actually resides. Not shown in the figure is the protection-enforcement portion of the memory management hardware. Its task is to verify the legitimacy of the intended mode of access to the target address before allowing the transaction with memory to proceed.

For items not present in main memory, the address-translation hardware generates an exception—or page fault—upon encountering a cleared (N) presence bit in the target PMT entry. The operating system maintains this "not present" bit to reflect the memory residence status of the related page. Upon demand, the operating system brings missing items into main memory from the secondary storage. Their addresses are stored in the file-map table (FMT), which is parallel to the PMT. Since the disk accessing needed to fetch a missing item is relatively lengthy, the operating system usually suspends the faulting program. It schedules another one for execution in the interim.

management schemes lies in the indirection provided by the page-map tables. The known static structure of these PMTs enables system designers to speed up table traversal by casting it in hardware. At the same time the operating system can furnish and update table entries—mapping descriptors—dynamically at runtime. Figure 1 on page 74 illustrates the mapping process by depicting two-level-deep, hierarchical PMTs in a paging system.

Hierarchical PMTs are sometimes called the address-translation tree. As Figure 1 shows, the root pointer to the translation table for each task is usually stored in the task control block. The TCB is an operating-system structure for task-related record keeping. Parts of the virtual address provide offsets into the corresponding levels of the address-translation tree. For instance, the root pointer and the level-1 part are combined to access the proper entry of the first-level table. It, in turn, contains the base of the appropriate second-level table to be indexed by the level-2 offset. This process is repeated until it reaches the last level of hierarchy (leaf node). Therein lies the page-map table entry with a physical page-frame address. Its contents are used to complete the address translation shown in Figure A in the box.

An attractive aspect of dynamic address binding and runtime address translation is that address spaces of partially executed tasks may be dynamically relocated in physical memory. This is accomplished by simply updating the translation tables and leaving the virtual

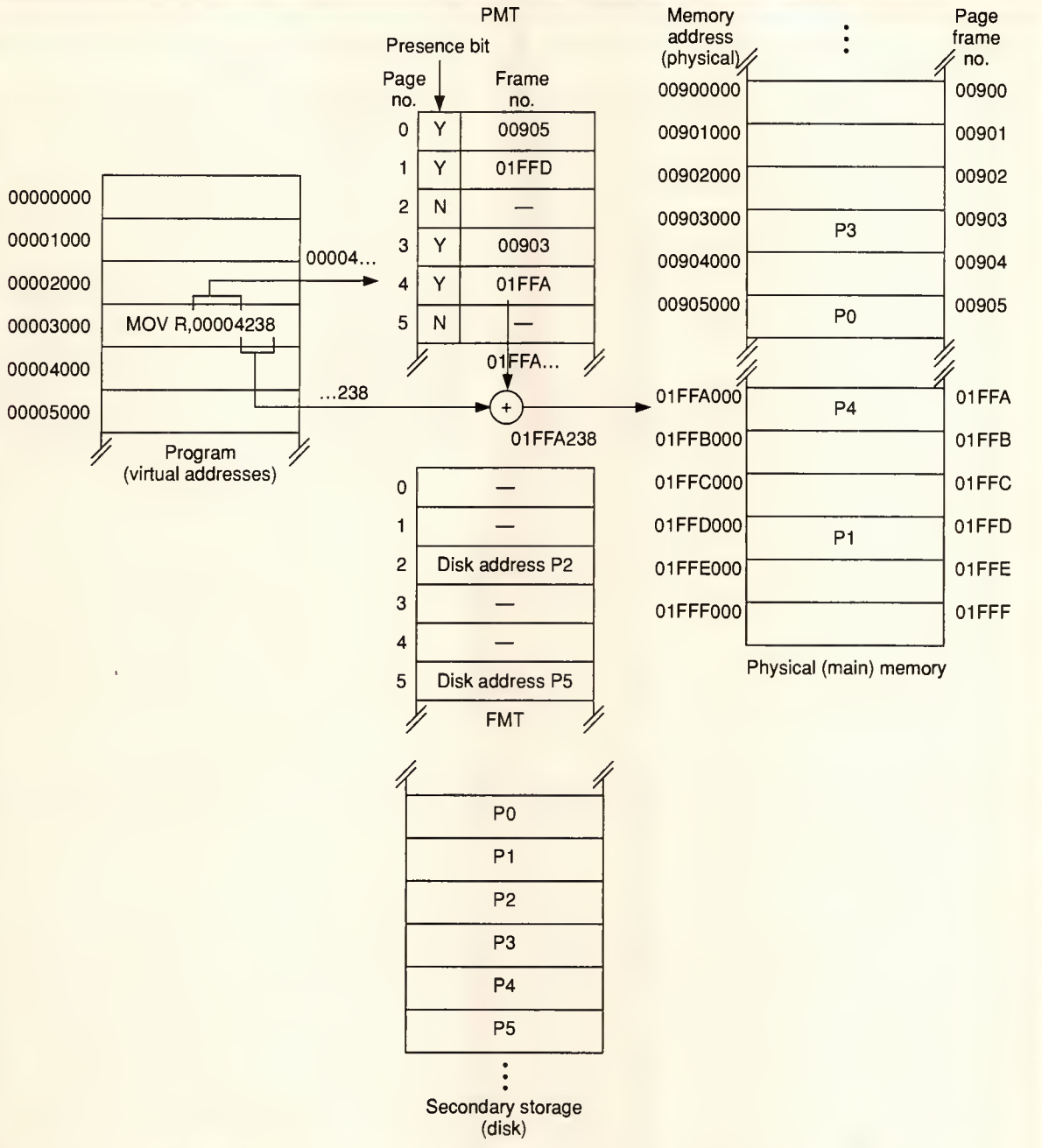


Figure A. Address translation in demand-paging, virtual memory systems.

addresses unchanged. As a side effect of address translation, the memory management circuitry enforces separation of distinct address spaces and supports controlled sharing of physical memory.

Isolation is enforced by separating distinct address spaces in physical memory and by ensuring that address translations map each task's virtual addresses strictly into its designated areas of physical memory. Access control bits stored in page descriptors provide the basis for protection and access control both within and between address spaces. Hardware uses access control bits during address translation to verify that the issuing task is authorized to access memory in the manner intended.

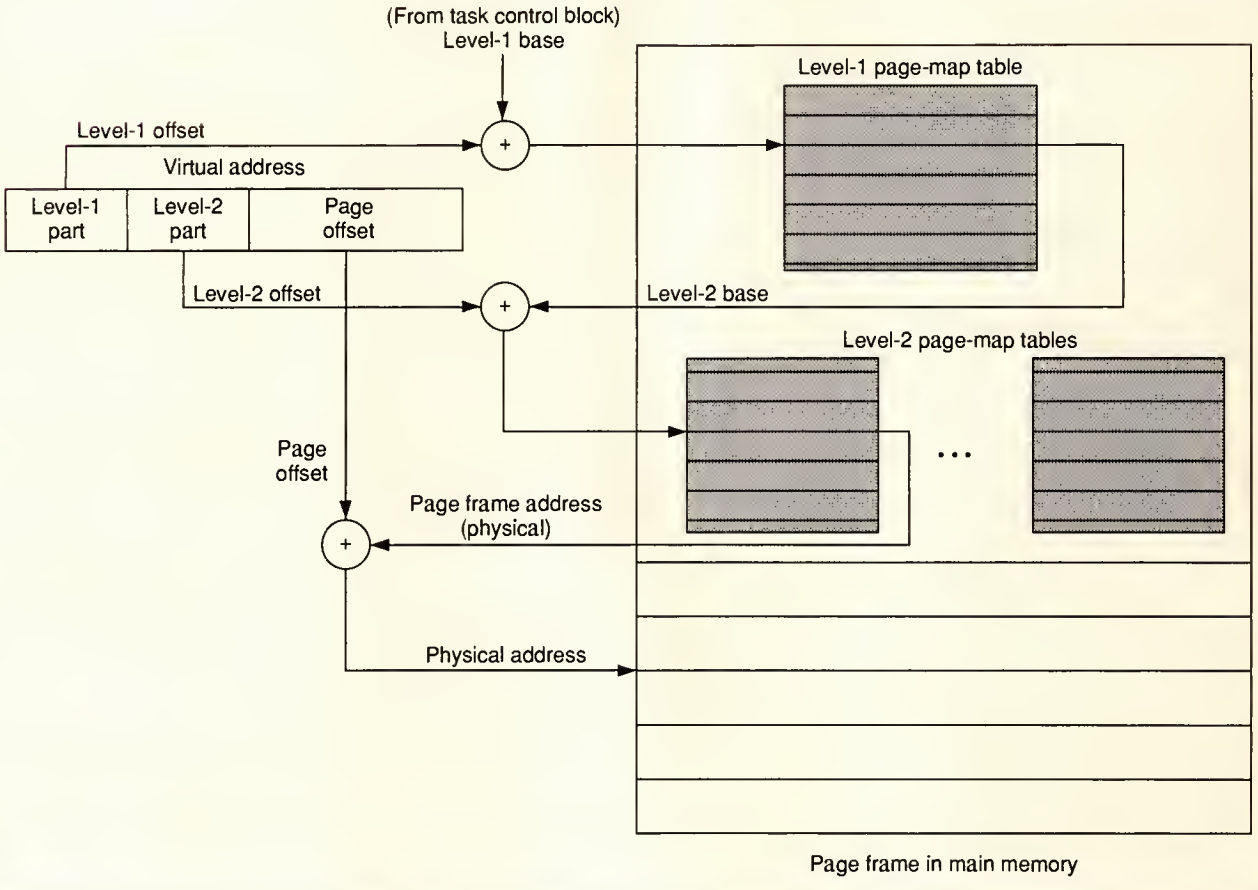


Figure 1. Hierarchical page-map tables.

Sharing of memory, when desired, is usually accomplished by having the operating system set the mapping tables so that they direct relevant virtual addresses of the participating tasks to the single physical copy of the shared item.

In virtual memory systems, the address-translation complex in virtual memory systems must also detect references to items that are not present in main memory and generate missing-item exceptions. The operating system subsequently processes missing-item exceptions by fetching the required items from auxiliary storage, typically a disk. Depending on the basic underlying memory management scheme, the described mode of operation is called demand paging or demand segmentation. If no free space is available in main memory to receive an incoming item, a resident item must be evicted to the backing store to make room. An item-replacement policy carried out by the operating system governs the choice of victim.

The price of this versatility is the complexity and overhead of runtime address translation. Depending on the underlying scheme—and with all items and mapping tables in main memory—it takes one or more memory accesses (references) just to translate a single virtual reference into the corresponding physical address. In other words, memory-based address translation reduces the effective memory bandwidth by 50 percent or more as compared to the same hardware with memory management turned off. Hardware assists in the form of descriptor caching, and fast translation buffers can alleviate this negative impact by reducing the number of main memory references expended on translations.

Memory management units. High-end microprocessors typically provide address-translation logic and hardware assists as parts of their MMUs. Since the MMU maps virtual addresses to physical addresses, it must be placed between processor-emitted (virtual) address lines and the physical-memory address lines and decoding circuitry. The MMU itself may reside on a separate chip, such as the 68851, or it may be integrated into the CPU chip so that it can output physical addresses directly onto the processor address pins. The types of instruction and data caches, often incorporated into the memory hierarchy of high-performance sys-

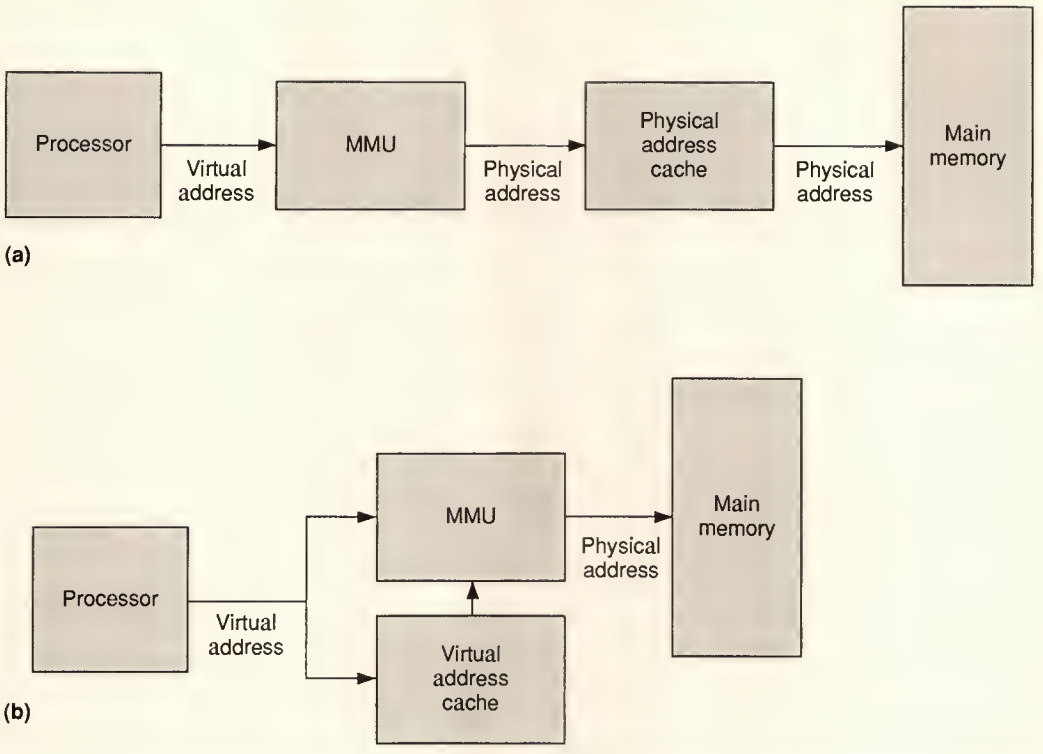


Figure 2. MMU and cache placement in physical (a) and virtual (b) paths.

tems, also influence this design decision.

Physical caches, prevalent in existing designs, operate with physical addresses obtained after translation in the MMU. Virtual caches, on the other hand, operate with virtual addresses. The primary advantage of virtual caches is the elimination of address-translation overhead when the target item resides in the cache. The drawback of virtual caches stems from the fact that virtual addresses are unique within an address space, but not between address spaces. As a result, virtual caches have to deal with the *synonym* or *aliasing* problem of recognizing all virtual addresses that map to the same physical memory address.

Figure 2 illustrates the placement of physical and virtual caches in processor-memory paths. A virtual cache uses virtual addresses and consequently must reside either on the processor or MMU chip. However, a physical cache operates with physical addresses and does not force on-processor integration of the cache when the MMU resides on the processor chip. Integrated processor/MMU chips generally provide faster translation and potentially additional speedup because they overlap some parts of the address translation with the cache lookup. Physical caches thus provide potentially greater design freedom as larger caches may be incorporated into the system without imposing the real-estate constraints of the single chip.

Primary responsibilities of MMUs in virtual memory systems include

- address translation with the appropriate assists,
- support for isolation and sharing in the course of translations, and
- detection and hardware-level processing of missing-item exceptions.

The underlying memory management scheme—segmentation, paging, or a combination of the two—largely determines the structure of the address-translation unit and its related assists.

Segmentation, by virtue of subdividing an address space into logically related groups (segments), generally provides superior protection both within and between address spaces. Other advantages of segmentation include ease of sharing both code and data, support for dynamic growth of segments, and dynamic linking and loading (a Multics idea recently featured in the IBM OS/2 operating system). From the hardware-support point of view, segmentation is attractive because it lends itself to deterministic caching of active segment descriptors in registers. This eliminates the need for the usually costly translation lookaside buffer common in paging systems.

Since segments occupy contiguous regions of physical memory and are not fixed in size, unused fragments of memory tend to develop between segments after a series of allocations and deallocations. The individual fragments are usually too small to receive a typical segment and thus remain unallocated. However, the

sum of these holes, that is, the total fraction of memory unused due to fragmentation in a steady state, can become quite large. This phenomenon, called external fragmentation, is one of the primary problems of pure segmentation. To alleviate the adverse effects of fragmentation on memory utilization, the memory manager must occasionally resort to costly compaction. This process usually requires relocation of resident segments into one end of memory to compact the fragmented free space. Variability of segment sizes also complicates allocation of both primary and secondary storage.

Paging mechanically splits each address space into fixed-size pages and allocates physical memory in same-sized quanta called page frames. The strength of this scheme is that it can load pages belonging to a single logical object—such as an address space of a task—into noncontiguous page frames in physical memory. Address-translation logic maintains the contiguity of virtual addresses that is required for proper program execution (see the example in the previous box). Immunity to the fragmentation of memory and the resulting elimination of compaction are the primary advantages of paged memory. The drawbacks of paging include less refined protection, and more difficult and restricted sharing of code—both relative to segmentation.

Both segmentation and paging require one mapping table lookup per level of hierarchy in address translation. This reduces the effective memory bandwidth by 50 percent or more in the absence of hardware assists. Segment descriptors may be cached deterministically by using segment typing to direct translation to the segment base register appropriate to the type of reference—code, data, or stack. As a result, the overhead of a mapping table lookup can be reduced to only a few percent of address translations at a modest cost of one caching segment register per type of segment. With no more than six or so segment types supported by most contemporary segmented architectures, the cost of caching active descriptors is quite low.

Unlike segmentation, pure paging does not observe logical relationships between items when placing them into physical memory. Because they are mechanically divided into pages, logically related items may be scattered in physical storage. One instruction can split between two noncontiguous pages. A single page can contain a mixture of logically dissimilar items such as a portion of code and a piece of stack. As a result, hardware assists in paging generally cannot benefit from the determinism in caching that segmented architectures make possible.

A common approach to reducing the overhead of address translation in paging systems is to use the descriptor cache better known as the *translation lookaside buffer*. The TLB contains associative memory in which it stores recently used page descriptors. Figure 3 summarizes the address-translation process in a system with a TLB, cache, and hierarchical PMTs. The TLB contains the corresponding (virtual address/PMT entry) pairs used by recent address translations. The address-translation hardware searches the TLB associatively by the page portion of the virtual address when the processor initiates a memory reference. When the match is found, the TLB furnishes the corresponding PMT entry. The physical address stored in the PMT entry forms the physical memory address. Protection is enforced by comparing task boundaries and access-rights bits stored in the PMT entry with the obtained address and the intended mode of memory access. The MMU flags discrepancies as protection violations and aborts the memory reference. In addition, a processor exception is raised to allow the operating system to abort the offending task.

TLBs generally have high hit ratios, in excess of 90 percent, and thus complete most address translations quickly. When a virtual address is not found in the TLB (a miss), the memory management hardware has to reference the PMTs in memory to complete the translation. The root pointer to the address-translation tree of a specific task is provided by the operating system as a part of the task-control block. The root pointer provides the base address of the next-level mapping table. The MMU uses it to reference memory and to obtain the related PMT entry. The presence bit therein is inspected to establish whether the item that it points to is resident in (physical) memory. If not, the page-fault exception is raised, and the operating system fetches the missing item from secondary storage.

When the target item is in main memory, the MMU checks whether it is the leaf node of the translation tree. If not, the process is repeated for each level of hierarchy until the last level is reached. As Figure 1 and the MMU survey in the next section indicate, only the leaf level contains pointers to page frames in main memory. Other pointers are indirect and serve as trail markers within the address-translation tree. Once the last level of the mapping hierarchy is reached, the MMU stores the obtained PMT entry into the TLB for future reuse. If the TLB is full, it replaces an existing entry to make room for the incoming one. TLBs commonly employ the least recently used (LRU) replacement policy. Simpler alternatives, such as FIFO and random, may be cheaper to implement but have inferior performance.

Due to the nonuniqueness of virtual addresses between different address spaces, single-task TLBs must be flushed upon every task switch. The oncoming task must usually resort to loading the TLB with its entries through a succession of TLB misses. Some designers opt to alleviate this problem by using multitasking or multicontext TLBs. A multicontext TLB removes ambiguities by tagging each entry with a unique owner identity, usually that of the enclosing task. On translation, both the address and the owner identity must match for a given entry to produce a hit. In this way, the TLB need not be flushed upon task switches, and a re-

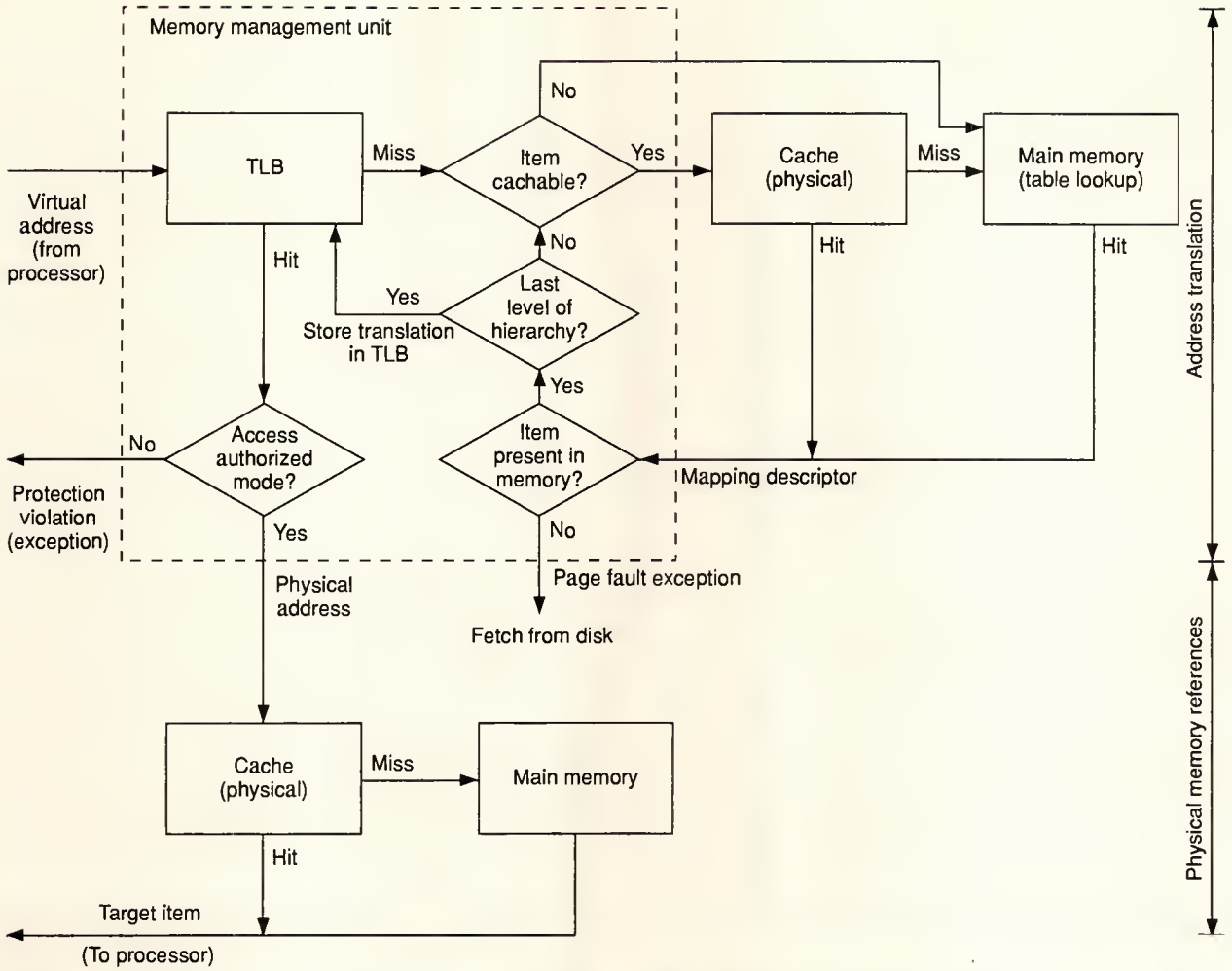


Figure 3. MMU and address translation.

scheduled task can benefit from finding some of its entries in the TLB.

Hardware support for virtual memory

The virtual-memory-specific aspects of address translation and MMU functionality include support for detection of missing items and hardware assistance to the operating system for page replacement. Most of these aspects come in the form of bits associated with each translation descriptor, such as segment and page descriptors.

One bit of each page-table entry is usually dedicated to indicate the absence or presence of the related page in main memory. This bit is often called *valid*, or *presence*. Software sets this bit to reflect when the related page is loaded into physical memory. Hardware uses the valid bit to detect missing pages while performing address translation. Another bit—called *modified, dirty*, or *written-into*—is usually dedicated per page-table entry to indicate whether the related page has been modified during its presence in main memory. This bit helps to reduce disk I/O by allowing the operating system to simply discard unmodified pages instead of writing them back to disk upon replacement. A third bit—often called *referenced*, or *accessed*—may be provided per PMT entry to assist the operating system in tracking page usage for replacement purposes. Hardware normally sets this bit when the related page is accessed. Software clears the bit in the course of gathering statistics on specific page usage. Some architectures (the Digital VAX family, for example) do not provide the referenced bit. The proprietary VAX/VMS operating system uses a working-set-based page allocation and replacement policy that does not require the use of the referenced bit. Unix, however, employs a global page-replacement policy whose operation bene-

fits from the use of a referenced bit. Unix implementations on the VAX address this problem by resorting to a somewhat slower, software-assisted emulation of the referenced bit. Most MMUs surveyed here provide hardware support for manipulation of referenced bits.

The choice of page size, which is usually fixed in a given system, is an important consideration in paging systems. Page sizes in current implementations vary between 512 bytes and 4 Kbytes. In general, small page sizes can lead to better utilization of physical memory by tightly enclosing specific localities of reference, which are generally believed to be quite small. On the other hand, the larger page sizes reduce the sizes of the PMTs and increase the transport efficiency with contemporary disk drives and local-area networks (LANs). Network considerations are becoming important as workstations with virtual memory—diskless or with comparatively small disk capacity—increasingly rely on remote servers for paging.

Fetching a page from a disk or a remote server consists of two major components, setup and page transfer. The setup time includes searching and disk-access delays until the target sector is reached. In LANs, setup includes gaining access to the network. The setup cost is incurred on a page basis, and it is largely insensitive to page size. The page-transfer time, on the other hand, is orders of magnitude shorter, and it varies in direct proportion to page size. With current technology, the per-byte cost of page transport is lower for larger page sizes in which the fixed setup cost may be amortized over many bytes.

In general, this trade-off is technology dependent, and its outcome fluctuates as the price and performance of individual components change. For instance, early implementations of paging favored larger page sizes mostly because of hardware-support considerations. More recently, the VAX architecture adopted a small page size of 512 bytes, which was probably influenced by the desire to make the best use of a relatively expensive RAM. On the other hand, large page sizes characterize contemporary implementations of Unix. This development probably occurred as a response to the current state of relatively abundant and fast memories, while disk and network access times have improved only incrementally.

Unix considerations

The Unix operating system has a memory model whose implementation can be facilitated by some specific MMU provisions. These provisions include support for discontiguities in the virtual address spaces, sparse address spaces, and tracking of page-usage history.

Unix has a notion of a *region* (called a segment in some early versions and in Berkeley releases) that is a contiguous area of virtual address space. The system can treat this area as a separate distinct object to be shared or protected. In most virtual memory implementations of Unix, regions are divided into pages. The address space of a task—which is a process in Unix terminology—consists of three separate sections: text (or code), data, and stack. While each section occupies a contiguous area of virtual memory, separate sections belonging to a single task do not have to be contiguous in virtual memory. This division, among other things, facilitates task management. For example, when a Unix task creates a child task via the fork directive, the two share a single copy of the text region. However, the child obtains a fresh private copy of the parent's data region. To save the overhead of copying a potentially large data area, some implementations of Unix let the two tasks actually share a physical copy of the data region until one of them modifies some data. At that time, the system obtains a new physical page frame and makes a separate physical copy of the original page prior to modification for the other party. The process of incremental page copying on an as-needed basis is often referred to as "copy on write." This approach reduces the copying overhead and conserves memory. However, it requires a somewhat more complex implementation of the PMTs.

Regions, discontiguities in virtual address space, and copy on write are much more efficient to implement with the hierarchical mapping tables shown in Figure 1 than they are with the more traditional linear, or flat, mapping tables. In many Unix implementations, a per-process region table contains pointers to the per-region page tables in which the actual page descriptors are stored. For example, parent and child processes sharing a region need only maintain private first-level pointers to a shared copy of the second-level page table in which the page descriptors are stored.

The resulting indirect mapping has some attractive advantages. Owners can have different access rights to a shared region as determined by their private first-level region pointers. At the same time, having one physical copy of the actual page descriptors greatly simplifies management of the physical storage. This eliminates multiple-copy consistency problems and localizes updates—such as those following a page swap or relocation—to a single point. Copy on write can be efficiently implemented when parent and child tasks share one physical copy of the data segment but maintain separate mapping tables to it. Other advantages of hierarchical mapping tables include efficient support of sparse address spaces and potentially smaller table sizes when the system maps discontiguous virtual spaces. Both of these advantages are relative to linear mapping tables.

A major disadvantage of hierarchical mapping tables is the slower address translation that results from looking up the intermediate pointers to locate the actual page descriptors. The added overhead basically amounts to one memory reference per level of indirec-

tion for each address translation. Hardware assists to partially offset the problem may come in the form of dedicated registers that hold the first-level region pointers for the executing task. In principle, TLBs in systems with hierarchical mapping tables should provide comparatively higher hit ratios to offset the longer translation times incurred by TLB misses. Increasing the number of entries, which also tends to increase the price, generally accomplishes a higher hit ratio.

Unix uses a global page-replacement algorithm that works better with the maintenance of longer page-usage histories. It is therefore convenient for Unix implementations to have MMUs with support for the referenced bit and provisions for storing the history of its changes into PMT entries.

Multiprocessor considerations

In multiprocessor systems with shared virtual memory and multiple MMUs, a mapping descriptor can be simultaneously cached in multiple TLBs. Since the contents of a mapping descriptor can vary in response to changes of state in the related page in memory, multiple copies of a given descriptor may pose a consistency problem. Keeping all copies of a mapping descriptor consistent with each other is sometimes referred to as *TLB coherence*. In a milder form, the coherence problem exists even in uniprocessor systems in which the TLB and memory copies of a mapping descriptor should be identical. Moreover, the updating of a memory-based descriptor should be an atomic action that makes the descriptor consistent with itself. The TLB coherence problem shares many characteristics with its better known cache-coherence counterpart.

As with caches, a crude way to deal with TLB coherence is to disallow TLB buffering of shareable descriptors. Forcing all mappings to complete by using the in-memory copy of a shared descriptor avoids the coherence problem at the cost of increased mapping delays. Numerous other ways to deal with the problem require some sort of hardware assistance such as snooping TLBs, remote TLB entry invalidation, and selective disabling or flushing of TLB entries. In systems that have both caches and TLBs, the two coherence problems are interdependent in perhaps nonobvious ways. For example, disallowing placement of shareable entries into TLBs may not achieve TLB coherence if caching of the mapping descriptors can occur and cache coherence is not enforced.

Consistency and coherence of shared entries in combined MMU/cache/memory systems still pose an open research problem. If the past is any indicator, some time will elapse before a few dominant solutions emerge and find their way into commercial implementations of cache and MMU units. In the meantime, designers of multimicroprocessor systems will have to resort to custom solutions.

Specific MMUs

This section describes the Intel, Motorola, Sparc, and MIPS MMUs. Table 1 on the next page provides a tabulated summary of their characteristics.

The Intel 80386. This processor is the first true 32-bit member of the iAPX 86 family. The upwardly compatible 386 contains a superset of the 8086/8's registers and instruction set, as well as the 286-compatible, on-chip MMU based on segmentation. The 386's prominent memory-management-related features include the removal of the annoying 64-Kbyte segment size limit (raised to 4 Gbytes in the 386) and support for paging suitable for the Unix-style implementation of demand paging. Its dynamic emulation of multiple, virtual 8086 machines may be of interest to designers involved with DOS-compatible multitasking.

The 386 can support up to 16K segments of 4 Gbytes each, thus allowing for single-task virtual address spaces of up to 64 terabytes (2^{46}). Since only 32 bits are provided on the address pins, the 386 can address 4 Gbytes of physical (real) memory.

The 386 provides extensive support for segmentation, which is inherent in its architecture. Strong segment typing and support for the four family-wide segment data types—code, data, stack, and extra, which are similar to Multics hardware design and recommendations—have been extended by two more data segments. Presumably, this addition should facilitate more extensive sharing of data without incurring the segment switch overhead. Segmentation provides for protection, sharing, and dynamic relocatability of code. Inclusion of the valid bit (called present) and of the reference bit (called accessed) into all segment descriptors allows implementation of the demand-segmentation style of virtual memory.

A major novelty in the 386 is support for paging. Unlike segmentation, paging is optional. When paging is enabled, the 386 actually supports combined segmentation and paging. This is the most powerful memory management scheme that exists in terms of protection and physical memory utilization. However, without hardware assistance, it requires three mapping memory references per each data reference. Segment typing and deterministic caching can virtually eliminate one mapping reference per translation. This leaves the TLB to cope with page-related mapping references in order to further reduce the loss of effective memory bandwidth. The 386 uses both techniques to improve performance.

The processor supports a fixed-size page of 4 Kbytes. A hierarchical, two-level structure of PMTs, a *page directory*, and *page tables* (in Intel terminology) keep

Table 1.
Summary of MMU characteristics.

Feature	80386/i486 Integrated	i860 Integrated	68020/030 68851 MMU/Integr.	68040 Integrated	88000 88200 MMU	Sparc MB86920 MMU	R2000/ R3000 Integrated
Virtual address space (Gbytes)	65,536	4	4	4	4	4	4
Physical address space (Gbytes)	4	4	4	4	4	64	4
Segmentation	Yes	No	No	No	No	No	No
No. of segments	16,384	—	—	—	—	—	—
Segment size	4 Gbytes	—	—	—	—	—	—
Paging	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Page size	4 Kbytes	4 Kbytes	256 bytes-32 Kbytes	4,8 Kbytes	4 Kbytes	4 Kbytes	4 Kbytes
Hierarchical tables	2 levels	2 levels	Up to 5 levels	Up to 3 levels	2 levels	Up to 3 levels	No
Table-walk hardware	Yes	Yes	Yes	Yes	Yes	Yes	No
Combination, segmentation and paging	Yes	Isomorphic	Isomorphic	Isomorphic	Isomorphic	Isomorphic	No
Protection							
Segmentation	Yes	Isomorphic	No	No	Isomorphic	No	No
Paging	Yes	Yes	Yes	Yes	Yes	Yes	Partial
Protection levels	4	2	MMU: 8; 030: 2	2	2	2	2
Virtual memory support							
Valid bit	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Modified bit	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Referenced bit	Yes	Yes	Yes	Yes	Yes	Yes	No
TLB (paging)							
No. of entries	32	64	MMU: 64; 030: 22	128 (64 + 64)	64	64	64
Associativity	Set, 4-way	Set, 4-way	Full	Set, 4-way	Full	Full	Full
Multicontext	No	No	MMU: yes (8); 030: no	No	No	Yes (256)	Yes (64)
Selective flush/lock	—	—	MMU, yes; 030: —	Partial	—	Yes	Partial
Probe	No	No	No	No	Yes	Yes	Yes
Multiprocessor (MMUs) support	386, no; i486, some	Some	Some	Some	Some	Some	Some
Cachability control	386, no; i486, yes	Yes	Yes	Yes	Yes	Yes	Yes

PMT sizes within reasonable bounds. This technique also provides efficient support for Unix-style sparse address spaces at the expense of an added level of indirection in the address-translation process.

Page-level access control bits (read/write) further expand the extensive protection and sharing capabilities offered by segmentation. Common entries in page directories allow sharing of page-table entries.

The system provides a full complement of valid (present), modified, and referenced bits at the page level. In addition, the 386 provides 3 unused bits to the operating system in each page descriptor for such purposes as history recording, copy on write, and "locked for I/O." The latter refers to locking pages in memory while they are involved in I/O operations such as DMA.

The 386 contains a four-way, set-associative TLB

with 32 entries. Because the TLB is single context, it must be flushed upon context switches.

The Intel i486. This machine is essentially a highly integrated, optimized 386 processor with on-chip memory management, a floating-point unit, and a cache memory unit. In addition to incorporating what used to be a separate 80387 arithmetic coprocessor, the i486 contains an 8-Kbyte, four-way, set-associative cache with a write-through policy. An interesting addition to the i486 that is gaining popularity in microprocessors of recent vintage is software-selectable byte-ordering in memory. In this way, the processor can support both little-endian (low-order byte first) and big-endian representations to ease the burden of format conversions in heterogeneous distributed systems.

The i486 MMU is fully compatible with the one found in the 386. The on-chip MMU supports segmentation, paging, and a combination of the two schemes. While segmentation cannot be turned off as in the 386, setting all segment base registers to zero provides a large, flat, address-space model. Probably for reasons of OS/2 binary code portability, the designers of the i486 stress its full support for the 286 style of segmentation. This compatibility is upward only, because the 286 does not support paging.

Two differences between the i486 and the 386 are page-level control of cachability and a somewhat improved, TLB, entry-replacement algorithm identical to the one used by the on-chip cache. The i486 uses 2 bits of each page descriptor (unused by the 386) to control the cache write policy, write through or write back, and cachability of the related page. The i486 also copies these bits to the outside pins to allow control of off-chip caches. The capability to declare certain pages noncachable and thus force their residence only in main memory provides a way to avoid the coherence problem in multiprocessor systems.

The Intel i860. This RISC-style, 64-bit processor has an on-chip MMU, a floating-point unit, and a cache unit. It also has an impressive declared performance rating of 40 MIPS (million instructions per second) and 80 Mflops (million floating-point operations per second), both at a 40-megahertz clock frequency. The chip contains a 12-Kbyte cache organized into separate 8-Kbyte data and 4-Kbyte instruction caches. It can operate as a stand-alone processor or in conjunction with a 386/i486-type processor.

The MMU of the i860 parts with company tradition by not supporting segmentation. However, the full paging portion of the i486 MMU is incorporated into the i860. The formats of the page directory, first-level mapping table, and PMT entries are identical in the i486 and i860, including support for the two cache control bits. In addition, the i860 contains a larger 64-entry TLB that is four-way associative. However, without segmentation, the i860 possesses a generally inferior, page-based, hardware protection scheme. Yet, because it maintains access control bits in both levels of the page-mapping hierarchy, the i860 allows more finely graduated forms of protection than many other paging MMUs.

The Motorola 68851 MMU. This stand-alone MMU, designed primarily for use with the 68020, connects to the processor through its coprocessor interface. Like most separate-chip MMUs, the 68851 offers richer functionality and suffers from longer address translation times than integrated, on-chip, MMUs. Although it offers a rudimentary form of segmentation, the 68851 is primarily oriented toward support of virtual memory based on demand paging.

The 68851 supports virtual address spaces of up to 4 Gbytes. With the use of function codes emitted by the processor, it achieves this value for each of the four distinct logical spaces: user code, user data, supervisor code, and supervisor data.

The virtual address space is logically subdivided into pages. The page size is program-selectable from a set of eight values ranging from 256 bytes to 32 Kbytes. The variable, programmable level of nesting in the address-translation tree forms a hierarchy of up to five levels deep. Thus, a 68851-based system can take up to five memory references to process a TLB miss. Page descriptors themselves can be long (8 bytes) or short (4 bytes), depending upon whether they include the size of the described object. Long and short descriptors can be freely mixed in a system, provided that all descriptors within a PMT are the same length.

An unusual 68851 feature allows its PMT to use indirect descriptors at the leaf level for pointing to another descriptor in memory rather than to a page frame. In this way, one physical copy of the page table descriptor can describe pages shared by several distinct address spaces. Efficient tracking of the state of shared pages results because only one physical copy of the related page descriptor needs to be updated. Alternatively, finding and updating all aliases of a given shared page whenever it changes state as a result of being modified or referenced would incur significant runtime overhead.

The protection mechanism of the 68851 is built on top of the paging mechanism and adds the refinement of user/supervisor access bits for each descriptor. Designers went to great lengths to compensate for the inherently inferior protection in paging systems by providing a hierarchy of up to eight levels of protection. However, the use of this feature is availed in conjunction with the rather complex and involved module call and return instructions (a CISC indulgence). Otherwise, the protection mechanism consists of the user/supervisor modes and the page-level, read/write access control for each mode.

The 68851 provides the standard complement of valid, modified, and referenced bits per descriptor.

The 68851 contains a relatively elaborate TLB called the address-translation cache. The TLB has 64 entries and a fully associative organization. Because they are distinguished by an associated task alias, entries belonging to up to eight distinct tasks can simultaneously reside in the TLB. For an address translation to succeed, both the table descriptors and the associated task alias must match those of the running task. In principle, such multitasking TLBs alleviate cold cache-start delays upon a task switch. Moreover, TLB entries can be locked to avoid replacement and thus speed up the translation of time-critical tasks at the expense of reducing the number of entries available to other users. The 68851's versatile TLB flushing allows selective purging of entries associated with a given task. This flexibility makes the TLB more responsive when processing task terminations or prolonged suspensions. Moreover, the system can preload the TLB with a set of descriptors belonging to a newly dispatched or activated task.

Multi-68851 configurations are supported by means of a cache-inhibit line that prevents caching of so-designated descriptors. Another facet of multi-68851 support is the provision for indivisible read-modify-write updates across the shared bus.

The Motorola 68030. This processor is the first member of the 68000 family with an on-chip MMU. It is an optimized version of the 68020 with separate address and data buses and caches. It also contains a proper subset of the 68851 MMU. From the memory management point of view the 68030 is interesting because—due to silicon limitations—it incorporates what its architects probably consider to be the essential features of the 68851 MMU. Since the architecture of the 68851 has been described, this section highlights only the differences between the two implementations.

The 68030 supports a 4-Gbyte virtual address space managed in the form of pages whose size is selectable in the range from 2^8 to 2^{15} bytes. Page maps and tables follow the same five-level, tree-structured hierarchy with a provision for early termination. The processor supports long and short forms of descriptors, as well as descriptor indirection, to facilitate the sharing of pages.

The 68030 has a provision for defining two large windows of memory that are not translated, a feature not offered by the 68851. This capability to escape address translation and its related overhead can provide efficient management of certain I/O devices such as bitmapped graphics.

The only type of protection available in the 68030 is through the use of the function code and the supervisor/user bit in the descriptors. The eight-layer, module-related form of access control available in the 68851 does not appear in the 68030.

The 68030's TLB is reduced to only 22 fully associative entries with a pseudo LRU replacement algorithm. The 68030 does not support multitasking and task aliasing. As a result, selective TLB flushing is not available. Thus all TLB entries in the 68030 belong to a single task, and the TLB is flushed in its entirety upon each task switch.

Motorola 68040. This processor is the current high-end member of the 68000 family. The chip is an optimized 68030 with an on-chip MMU, a floating-point unit, and two cache memory units. The Harvard-architecture-based chip separates instruction and data buses and maintains a separate cache for each. As a major innovation, the MMU incorporates two separate TLBs, one for instructions and one for data. The underlying motivation for this design could be that instructions and data tend to have separate loci of activity that dedicated TLBs can capture more efficiently. Each TLB is four-way set associative and contains 64 entries, for a total of 128 stored instruction- and data-page translations. The processor does not support multiple contexts, and TLBs can mix only supervisor entries with single user-task entries at a given time.

The MMU supports two software-selectable page sizes of 4 and 8 Kbytes. PMTs contain a hierarchy called an address-translation tree. The hierarchy can comprise up to three levels with a provision for leaf-level indirection as in the 68030. In line with the 68000 family tradition, the system can maintain separate address-translation trees for user and supervisor (operating-system) address spaces. Two dedicated registers in each TLB allow direct accessing of two memory regions without the benefits and overhead of address translation.

Individual PMT entries, also called page descriptors, contain support for valid, modified, and referenced bits. Page descriptors also contain provisions for recording the referencing history and controlling page-level cachability. The latter is rather elaborate in the 68040, with specific modes for write-through and copy-back policies. In addition to the noncachable option, a specific mode enforces serialization of memory references as dictated by the instruction order and thus circumvents inversions that might result from pipeline or copy-back schemes.

The Motorola 88200 MMU. The 88200 is a one-chip combination of a cache and an MMU (called the CMMU) for the 88000 family of RISC-style processors. In these system configurations, up to two 88200s per 88100 processor can serve as dedicated instruction and data caches. The instruction CMMU also serves as a memory manager. The CMMU adds convenience to 88000 designs by providing the memory-bus interface and thus frees some of the processor chip real estate. Combining the cache and the MMU allows address translation and cache-tag matching to proceed in parallel.

The CMMU contains local intelligence in the form of a finite state machine that can perform address-transla-

tion table walks without processor assistance. As in the 68851 and 68030, the processor's only task is to produce a virtual address. The CMMU handles all other aspects of address translation except for page faults. On cache hits, the 88200 can furnish up to one word of memory per 88100 processor cycle. When it communicates with memory, the 88200 attempts to employ 4-word (32-byte) transfers whenever possible to make the best use of the burst mode and to exploit characteristics of static-column dynamic RAMs.

The 88200 supports virtual memory based on demand paging. PMTs contain a fixed, two-level hierarchy. The first-level mappings are called segments as in Unix terminology.

The CMMU supports a virtual address space of 4 Gbytes. With the added refinement of the mode bit, two such spaces—user and supervisor—can exist simultaneously. The page size is fixed at 4 Kbytes. Page tables are organized as a fixed, two-level hierarchy of 1,024 segments, each with up to 1,024 pages of 4 Kbytes. Second-level PMT entries have a format compatible with the 68851 and 68030 to facilitate sharing of page tables in hybrid systems composed of 68020s, 68030s, and 88000s.

As is the case with the 68030, protection in the 88200 rests on paging coupled with a mode to differentiate between user and supervisor access rights at the page level.

The CMMU hardware provides and manages valid, modified (or used), and referenced bits in a standard fashion.

The TLB, which the company calls the PATC (page address-translation cache), has 56 entries and a fully associative organization. Multitasking is not supported, and all entries apparently belong to a single task. Consequently, the cache must be flushed upon each context switch. In addition to the usual complement of TLB operations, the 88200 includes a provision for probing the PATC, that is, for initiating translation without causing an exception. This feature can be used to precheck for page faults and thus indirectly prolong residence of certain descriptors in the cache. The latter is a side effect of probing because probes, like translation, cause the referenced bit of the affected descriptors to be set.

An unusual feature of the 88200 is its management of fixed-size, 0.5-Mbyte, unmapped blocks of memory that are contiguous. A separate 10-entry, fully associative TLB called the BATC (block address-translation cache) is included in the CMMU for this purpose. Block-oriented memory is intended primarily for the operating system and its storage of PMTs. The idea is presumably to improve efficiency by eliminating paging and its associated overhead when the operating system executes or accesses its page tables. The BATC is searched in parallel with the regular, paged PATC. A hit in the BATC completes the translation by providing access to the related block in physical memory. A miss in the BATC causes inspection of the outcome of the PATC search. A hit therein provides page-based translation. Finally, a PATC miss causes the CMMU to walk through the mapping tables in the memory-segment descriptor, followed by a page table lookup (one memory reference each). The address obtained in this manner references the target item in memory, and the table-walk trail is stored in the PATC for future use.

The 88200 incorporates a rather sophisticated cache controller with provisions for multicache coherence such as bus snooping and intention-to-modify bus transactions with synchronization.

The Fujitsu MB86920 MMU. Sun Microsystems has commercialized a RISC-style chip based on the RISC and RISC II developed at the University of California, Berkeley. Sun's version, called the Scalable Processor Architecture (Sparc), has been licensed to several semiconductor vendors that include Cypress, Fujitsu, and Texas Instruments. Besides the customary RISC assortment of pipelining, one-cycle instructions, a load/store architecture, and delayed branches, Sparc contains a large number of registers arranged into register windows that are a trademark of the Berkeley RISC.

An unusual aspect of the Sparc-based system design is that the MMU is not a part of the architecture. Sun uses a gate-array implementation of a proprietary MMU with a large, direct-mapped address-translation cache. Other vendors provide their own, often incompatible, MMUs. This situation may have an adverse impact on the binary compatibility of Sparc system software.

The Fujitsu MB86920 is a rather elaborate, separate-chip MMU intended for support of Unix and its derivative SunOS. In terms of complexity and functionality, the MB86920 seems closer to CISC MMUs than to their somewhat leaner RISC counterparts.

This MMU provides a Unix-inspired memory model based on paging with hierarchical mapping tables up to three levels deep. The MMU divides the Sparc-emitted virtual address into three fields that correspond to the Unix-like notions of region, segment, and page. The page size is fixed at 4 Kbytes. With its early-termination provision, the MB86920 can thus support linear mappings of sizes that range from 4 Kbytes to 1 Gbyte.

Each leaf PMT entry (descriptor) contains modified and referenced bits for the corresponding page. The descriptor also contains a 2-bit cluster that encodes the valid/invalid state and the type of the related page. A dedicated bit in each PMT entry provides cachability control at the page level.

The MB86920 stores recent translations in a 64-entry, fully associative, multicontext TLB. The MMU provides a variety of modes for probing and selectively flushing the TLB entries, including individual, context, and page-level flushing. The system can retain kernel entries while flushing the TLB.

Support for multiprocessor environments—in addi-

tion to page-level cachability control—includes a provision for automatically updating mapping-table entries in memory whenever the corresponding entry is updated in the TLB. When mapping tables are declared noncachable, this feature provides a simple but workable mechanism for TLB coherence in multiprocessor environments.

The MIPS R2000/R3000. This architecture directly descends from the Stanford University RISC project called MIPS (Microprocessor without Interlocked Pipelined Stages). The R2000 and R3000 processors share a common architecture, the latter being a faster implementation. Both processors support software-programmable, big-endian and little-endian byte-ordering in memory.

The MIPS approach is somewhat unusual in that it fully architecturally integrates the MMU. The MMU-specific registers and instructions are part of the basic processor design; the MMU cannot be switched off. For performance reasons, address translation may be bypassed for specific areas of the virtual address space whose address ranges are hardwired.

True to the RISC tradition, the MMU portion of the MIPS architecture provides only the basic mechanisms for virtual memory support and leaves the rest to software. MIPS designers envision rather elaborate high-performance system designs based on their processor (such as separate instruction and data caches with write-buffer speedup). However, they delegate most of the exotic functions to external hardware and system software. This approach probably results from the MIPS project philosophy of providing relatively simple hardware and relying heavily on software such as optimizing compilers for performance.

The MIPS on-chip MMU supports paging with fixed-size pages of 4 Kbytes. It is the only MMU surveyed here that does not provide hardware support for table walking. The MMU just signals translation (TLB) misses via processor exceptions and leaves accessing of page maps to software. Thus, operating-system designers have the freedom to choose their favorite table organization as well as the burden to implement it in software.

R2000/R3000 processors provide 64-entry, fully associative TLBs with support for multiple contexts. The bits supported by the R2000/R3000 MMU include valid, modified (or dirty), and per-page cachability control. Hardware does not support the referenced/accessed bit. A dedicated TLB exception facilitates probing.

The TLB does not provide an entry-replacement algorithm in hardware. Instead, the processor contains two specialized MMU registers called index and random. They facilitate implementation of software-assisted algorithms, including random replacement and replacement/flushing of a specific TLB entry.

Microprocessor MMUs provide hardware support for address translation coupled with registers and circuitry for enforcing protection and facilitating sharing between address spaces. In addition, all MMUs surveyed here support valid and modified per-descriptor indicators needed for implementation of virtual memory. All but one of the MMUs also support the referenced indicator that accumulates in-memory, item-usage statistics required by the item-replacement routines of operating systems such as Unix.

All presented MMUs support paging, some support segmentation, and only a few provide true combined segmentation and paging. Hierarchical page-map tables—useful for sharing, sparse address spaces, and implementing the Unix memory model—are becoming standard features in microprocessor MMUs. In addition to the basic two-level scheme, some MMUs support three or more levels with optional indirection to facilitate sharing. Early termination supports variable sizes of unmapped objects. In operational terms, hierarchical mapping—especially with a variable number of levels—provides many of the attributes normally associated with segmentation. However, unlike segmentation, the size of a target object must conform to one of the fixed sets of available sizes.

In MMUs that lack variable page size, fixing the page size at 4 Kbytes constitutes a clear consensus. Except perhaps for the MIPS, few if any significant differences exist between CISC and RISC MMUs. However, RISC designers appear to prefer simpler solutions, such as no segmentation and a less complex hierarchy of tables. Complex MMUs are probably undesirable in RISC systems. They can lead to processor slowdown and disruptions of the instruction pipeline due to potentially lengthy and uneven address-translation times.

The TLBs surveyed here are either fully associative or at least four-way set associative in organization. TLB sizes vary from 22 to 128 entries, with multiple contexts supported in some larger TLBs. These TLBs also tend to support selective entry flushing, and some even support locking of designated entries. Entry replacement gravitates toward hardware implementations of the pseudo LRU algorithm. Another discernible trend is the incorporation of cachability control into page-map table entries (descriptors). The resulting fine granularity of control is achieved at a modest cost of one or two dedicated bits per page descriptor.

In general, caches and MMU designs are becoming more related by virtue of the dependencies of their placement in the processor-memory paths and by concerns about cache and TLB coherence in multiprocessor systems. The units presented here support only physical caches and include some cache coherence but virtually no TLB coherence provisions. Proliferation of multiprocessor systems and the tendency to integrate caches and MMUs on the same chip will force greater consideration of these issues and probably spawn a variety of designs in future systems.

Acknowledgments

A grant from the NCR Corporation partially funded this work. Lee W. Hoevel of NCR provided the initial inspiration and much of the support for this project. The author completed this study while he was a faculty member at the Southern Methodist University in Dallas, Texas.

Bibliography

Agarwal, A., J. Hennessy, and M. Horowitz, "Cache Performance of Operating Systems and Multiprogramming Workloads," *ACM Trans. Computer Systems*, Vol. 6, No. 4, Nov. 1988, pp. 393-431.

Archibald, J., and J.L. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," *ACM Trans. Computer Systems*, Vol. 4, No. 4, Nov. 1986, pp. 273-298.

Bach, M.J., *The Design of the Unix Operating System*, Prentice-Hall, Englewood Cliffs, N.J., 1986.

Clark, D.W., and J.S. Emer, "Performance of the VAX 11/780 Translation Buffer: Simulation and Measurement," *ACM Trans. Computer Systems*, Vol. 3., No. 1, Feb. 1985, pp. 31-62.

Denning, P.J., "Virtual Memory," *ACM Computing Surveys*, Vol. 2, No. 3, Sept. 1970, pp. 154-189.

i486 Microprocessor, No. 220440-001, Intel Corporation, Santa Clara, Calif., 1989.

i860 64-Bit Microprocessor, No. 240296-002, Intel Corporation, 1989.

Kane, G., *MIPS RISC Architecture*, Prentice-Hall, Englewood Cliffs, N.J., 1988.

Milenkovic, M., *Operating Systems: Concepts and Design*, McGraw-Hill, New York, 1987.

MC68030 Enhanced 32-Bit Microprocessor User's Manual, MC68030UM/AD, Motorola Inc., Phoenix, Ariz., 1987.

MC68851 Paged Memory Management Unit User's Manual, MC68851UM/AD, Motorola Inc., 1986.

MC88200 Cache/Memory Management Unit User's Manual, MC88200UM/AD, Motorola Inc., 1988.

Smith, A.J., "Cache Memories," *ACM Computing Surveys*, Vol. 14, No. 3, Sept. 1982, pp. 473-530.

SPARC MB86920 High-Performance Memory Management Unit, Fujitsu Microelectronics, San Jose, Calif., 1988.

68040 User's Manual, (excerpts from the preliminary version of the manual scheduled for release in 1990, courtesy of Motorola), Motorola Inc.

80386 Programmer's Reference Manual, No. 230985-001, Intel Corporation, 1987.

80386 System Software Writer's Guide, No. 213499-001, Intel Corporation, 1987.

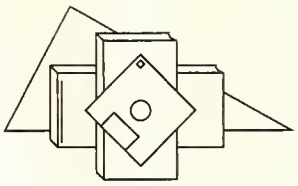
Milan Milenkovic is a member of IBM's advanced technology systems development group in Boca Raton, Florida. Prior to joining IBM, he was a faculty member in computer science and engineering at the Southern Methodist University in Dallas and the University of Massachusetts at Amherst.

Milenkovic's interests include operating systems, computer systems design, multiprocessors, and distributing processing systems. He authored a number of papers and technical reports, a monograph on database concurrency control, and a book on operating-systems design. The National Technological University (NTU) network broadcast his courses to industrial audiences.

Milenkovic received an MSc degree in computer science from the Georgia Institute of Technology and a PhD in electrical and computer engineering from the University of Massachusetts. He is a senior member of the IEEE and a member of ACM, Usenix, and the IEEE Computer Society.

Readers may address any questions concerning this article to the author at IBM Corporation, 1000 NW 51st St., IZIP 5226, Boca Raton, FL 33431.

Department



Micro Review

Richard Mateosian
Hitachi America, Ltd.
(415) 244-7456
Fax (415) 583-4207

Software quality tools

For as long as I can remember, the process of software development has been out of control. In the 1960s, high-level languages promised to make things better. With hindsight, it's easy to see why they made things worse. Developers could build larger systems in the same amount of time. Larger systems made the lack of specification, design, and prototyping tools more glaringly obvious than before.

Early in the 1970s, we began to use the term "software engineering" for what was previously called programming and systems development. Techniques for structuring programs and data caught on. People with little insight into the problems of systems design advocated these techniques and followed them slavishly. More significantly, structured approaches to code and data design crept into programming languages like Pascal, Modula II, and even C.

Because of the advances in coding and data structuring, most bugs in software projects now occur in the specification and design phases. These bugs are usually not detected until late in the project, and they are the hardest to correct. The tools to reduce errors in specification and design date from the 1970s, but they are only now evolving beyond pencil-and-paper techniques carried out by specialists. The proliferation of graphic interfaces has led to a category of programs called computer-aided software engineering (CASE) tools. In their present infancy they use the computer mainly to replace the pencil and paper of the previous generation. But the computer's "nose in the tent" promises qualitative improvements like direct compilation of specifications and designs into code.

This month I review two books in this area. The first updates a classic from 1977. The second surveys current CASE tools.

Structured Walkthroughs, 4th ed., Edward Yourdon (Yourdon/Prentice-Hall, Englewood Cliffs, N.J., 1989, 207 pp., $30.80, hard cover only)

A walkthrough is simply a peer-group review of a piece of work, usually some aspect of the design and development of a computer-based application. Yourdon introduced the term in the course of developing and teaching his well-known design methods. The idea goes back at least to Gerald Weinberg (*The Psychology of Computer Programming,* Van Nostrand Reinhold, New York, 1971). In practice it means that a programmer who has reached a predetermined milestone in developing a program makes the listings available to other programmers in the development group. Then they all go through the listings together, and the programmer notes the suggestions and subsequently integrates them.

Yourdon introduces walkthroughs and shows how to implement them in various situations, not just program development. He points out some of the psychological pitfalls that can occur and shows how to avoid or deal with them. Different chapters look at walkthroughs from the viewpoints of managers and technicians. The ones written for managers contain important insights for old-timers who find themselves managing groups that use newfangled techniques like walkthroughs. For example, walkthroughs need openness for their successful employment. Employees are aware that their performance evaluations and subsequent pay increases depend upon making good impressions on their managers. This awareness may inhibit the required openness, which is one rea-

0272-1732/90/0400-0086$01.00 © 1990 IEEE

son why managers are advised not to attend walkthroughs.

In a chapter added for this edition, Yourdon looks at the future of walkthroughs. CASE tools and rapid prototyping have changed the focus of these techniques, but the basic idea remains valid. In the final analysis, what was true in 1971 is true today. Organizations will realize the benefits of walkthroughs when their underlying cultures evolve away from the "cowboy programmer" paradigm and toward egoless "groupware."

CASE—Using Software Development Tools, Alan S. Fisher (John Wiley & Sons, New York, 1988, 303 pp., $24.95, paperback)

Skip the next two paragraphs if you are tired of hearing me carp about bad editing.

The job of an editor is a lot easier than it once was. The author submits material in machine-readable form. The editor can manipulate text electronically, and the author can make suggested structural changes relatively easily. Spelling checkers and similar tools eliminate much of the mechanical work of editing. The editor can concentrate on assuring adherence to the publisher's standards of excellence for form and content. In this book, however, the editor didn't do that job.

Fisher says in his acknowledgments: "Teri Zak ... deserves praise for shepherding this book through the review and production process. Her behind-the-scenes work made this all appear very transparent to me, much like a good CASE tool does with software." Unfortunately, the tool was inadequate. Fisher is left with his name attached to such glaringly bad English as "a software design principal," "pouring over the design specification," "complimentary methodologies," "the immediacy of . . . deadlines compel junior programmers to start," "the cause of most . . . problems stem from," and even "code geration." I took all of these examples from a span of 16 pages early in the book.

Obviously, these irritating examples have little bearing on the usefulness of the book. Fisher has tried to do four tasks, and each corresponds to a group of chapters. For the project manager, he reviews the classical model of the software development process and shows where CASE fits in. For developers and working managers, he surveys the manual techniques that are implemented in current CASE tools. For those burdened with selecting CASE tools, he details 10 current CASE products. Finally, for an audience I can't quite identify, he groups everything else he wants to say. These topics range from character traits of CASE tool users to DoD-Std 2167 to marketing strategies of CASE tool vendors.

I think that the detailed descriptions of 10 current CASE tools are likely to sell the most books. If you need to select CASE tools, then get this book. Once you have it, you might find some of the other parts interesting.

The Matrix

The productivity tools I just described are only one piece of the productivity-enhancement pie. Another "tool" that many of us use—but few understand—is the large, sprawling network that joins computer users of all degrees of sophistication from all around the world. The next book I review attempts to clarify this mysterious network. Like its subject, it is a work in progress.

The Matrix — Computer Networks and Conferencing Systems Worldwide, John S. Quarterman (Digital Press, Bedford, Mass., 1990, 746 pp., $49.95, paperback)

This tome is awe inspiring. I haven't read it; I've merely paged through it. If you do the same in a bookstore, I'm certain that you'll feel an overwhelming urge to buy it.

The book is a successor to "Notable Computer Networks," which the author published in the *Communications of the ACM* in October 1986. Its title derives from the science fiction novel *Neuromancer* by William Gibson.

The book divides into two parts. Part 1 comprises the first 213 pages. It basically consists of background material for Part 2, but most readers will find much of it extremely interesting. The discussion touches on everything from underlying technology to Matrix etiquette and ethics.

Part 2, which comprises more than 500 pages, describes specific networks and conferencing systems, their interconnections, and their uses. The author organizes them geographically and includes a number of maps. He also provides information about accessing these systems and sending mail between them.

Designed as a reference work, the book became obsolete before the first copy came off the presses. Consequently, the author has developed an interactive relational database of information related to the Matrix and actively solicits suggestions for features to include in an ongoing service. You may reach the author at The Matrix, PO Box 14621, Austin, TX 78761, matrix@longway.tic.com, Fax (512) 471-2449.

Department

Micro World

Hubert Kirrmann
Asea Brown Boveri Research Center
CRBC.1
CH-5405 Baden, Switzerland

Minitel: The French love affair with telematics

Calling the permanently overloaded telephone information service in France (dial 11) requires a lot of patience. Searching through 40 or more obsolete and vandalized telephone books in a post office to find a specific party is even more tedious.

In 1978, the French Postes, Telephones et Telecommunicacions (the PTT) recognized the problem and started an experimental electronic directory service, or Teletel. The establishment of such a telephone directory was not its primary motivation, however. The PTT was looking for ways to increase traffic, and it foresaw new, expansive domains such as providing access to railway timetables (to solve another notorious public problem). At that time, the PTT did not foresee it was laying the foundation for the first major breakthrough of public telematics—the French call it *telematique*—or the convergence of telecommunications and automatic information processing.

It all began in the city of Velizy with an experimental terminal connected to a telephone line that used a domestic TV set as a display. (Teletel belongs to the videotex class that employs telephone lines.) Researchers experimented with Teletel on a wider scale in Brittany near the PTT research center. Their self-contained terminal included a small, 23-centimeter, black-and-white screen. In an attempt to ease the terminal's acceptance, researchers organized the keyboard alphabetically rather than in the usual typewriter style.

In February 1983, the PTT launched the first full-scale Teletel service. The subscribers in the Ile-et-Vilaine area could choose a directory book or a terminal. Both were free. Those who chose the terminal—called Minitel—soon discovered the virtues of the electronic directory. They could search for a person by approximate name, street, region, or phone number. They could also locate the nearest dentist or carpenter. Within a few keystrokes, they could have a complete list of relatives in Paris.

Other services besides the directory developed quickly: listings of railway and airplane timetables, restaurants, popular events, taxi services, tourist offices, weather information, and horse races, as well as access to professional databases. The success of Minitel—as the public soon began to call the whole videotex—was immediate.

In December 1984, Minitel expanded to the north, the east, and the south coast of France, and in 1985 it covered the whole country. By the end of 1989, France Telecom—the organization that succeeded the PTT in view of the unified market of 1992—had installed 5 million Minitel terminals in one out of every three households in France (see the box).

How do you use the terminal? You connect it in parallel to the normal telephone (without disturbing phone service). To access Teletel, you dial a four-digit number on the phone and push the connection key when you hear the high-pitched sound of the modem, as with a telefax. A menu pops up. To call a service, you enter its acronym on the keyboard. The service announces itself.

One interesting aspect of Minitel is payment for services. The initial charge for a local call is about US twenty cents. Some services like the directory are free, at least for a limited time. The called party pays for some of the other services. France Telecom bills the remaining services on behalf of the called party according to a predefined scheme. The services now divide into seven categories, or kiosks, subdivided into invoice levels (see Table 1). Finally, many services require an additional subscription (with or without a fee) and a special password. The provider bills users a flat monthly fee or for the amount of time the device was actually used.

Although about 8,000 different services exist, only one third of them are really active. While the private person is most likely to use the directory, firms tend to access the stock exchange, airline reservations, libraries, or other databases. Chemical, pharmaceutical, automotive, and electronic firms open their catalogs and accept orders directly. Private networks exist within Teletel; some companies use Minitel to transmit maintenance instructions to the field.

Among Minitel's special services are mailboxes on which you can leave messages for others. These mailboxes exist for special interest groups like young-scientist organizations or chess clubs. *Messageries conviviales* (guest-messaging procedures) allow people to communicate in conference calls while preserving their anonymity. A participant uses a pseudonym known only to the server. At first, *messageries* were believed to fulfill a social function by putting lonely or timid people in contact with each other. The city halls originally started these services as social programs for elderly and socially marginal people. But soon these services were exploited as

0272-1732/90/0400-0088$01.00 © 1990 IEEE

Inside Minitel

The Minitel terminal is the world's most widespread, with a production rate of about 700,000 units per year. Telic/Alcatel makes about two thirds of the terminals, with the Matra Company ranking second in production figures. Telic/Alcatel manufactures the terminals in the city of Woerth in Alsace (east of France) at a rate of 2,000 units a day. Although the bottom-line Minitel terminal is free (instead of the usual directory books), many firms buy specialized multistandard terminals for about US$400.

The technical implementation of a Minitel terminal is nothing spectacular. It looks like a ruggedized version of a vintage 1980 Tandy TRS80. The screen displays only 25 lines by 40 columns. Its half-graphic display code is very similar to that of the TRS80 but also provides eight levels of gray, a color option, and, of course, the French character set. The baseline Telic/Alcatel Minitel comes with an 8039 microcontroller, a pair of Thomson EF 9340/9341 video controllers, and a Motorola 6850 serial interface. The tiltable, 65-key keyboard now features the classical typewriter layout because secretaries were upset by the alphabetical arrangement and inexperienced persons had to search for the keys.

The integrated modem operates according to the V23 CCITT transmission standard (1,200/75 baud). The 1,200-baud channel supports the display and the 75-baud channel supports the keyboard, but the channels can reverse for terminal-to-terminal linking. The Minitel has a simple, link-layer protocol. It provides connection services, terminal identification, and acknowledged frame transfer. It connects such peripherals to the terminal as a smart-card reader, printer, or PC. Up to four Minitel peripherals can be chained through a simple five-pin, Din (Deutsche Industrie-norm) audio connector without a repeater. The peripherals implement a multimaster protocol and communicate directly with the remote server. In contrast to terminal functions, they can operate at full speed in both directions if the medium allows it.

Several multistandard versions of the Minitel exist, allowing connection to the United Kingdom's Prestel or to the German Bildschirmtext, as well as to VT50, VT102, Bull, or IBM emulations. These increasingly intelligent terminals can dial by themselves directly from the directory pages, recall previous pages, and respond automatically. Printing is manually triggered. Some of these terminals have mass storage and are one step away from being a PC. In fact, you can go the other way around and buy cards and software for a PC to transform it into a Minitel terminal. This approach avoids having two screens on one table and opens a new dimension. Once you have mass storage, you can perform file transfer.

The Teletel network connects over the normal telephone lines to the nearest exchange. There, one modem out of a set of some thousand assigned to the line digitizes the data. It feeds a mainframe (such as CII Alcatel) that performs the initial dialogue, provides directory services, or forwards data to a remote service. The data transits over Transpac, which is the French packet-switching network. It is similar to Datex-P in Germany, Telepac in Switzerland, or Tymnet in the US. The server computers preferably connect directly to Transpac, but they may include additional modems to increase actual operation time through alternative routing.

Anybody can offer services on Teletel. All you need to do is buy a computer with the proper software, connect to Transpac, and ask for a Teletel acronym. Strict rules declared by France Telecom rule server implementation. These rules, available from Cnet (see address box on next page), define such things as the protocol, user interface, and paging modes. A server can also connect directly to a mainframe. In this case, the Minitel terminal should support a VT52 or similar computer terminal mode. Some firms offer preconfigured servers for *messageries* or catalogs.

the famous *messageries roses*. The exploiters affixed people's Minitel numbers to city walls along with evocative illustrations. France Telecom claims that it stops all mailboxes dealing with sex, drugs, or prostitution within seconds, but the truth is that the principle of conversational privacy hampers such attempts.

A rather annoying fact is that people use the *messageries* and other awkward services like emergency astrological help mostly during work hours. Private firms—upset by the phone costs and lost work hours—avoid installing lone Minitels. Phone bills have also increased sharply

Table 1.
Payment for services.

Kiosk No.	Type of announcement	Price (US$/hour)
3605	Green number (no charge)	—
3613	Paid by callee	$1.00
3614	Paid by caller	$2.00-$3.00
3615	Public kiosk	$5.50-$11.00
3616	Low-cost professional	$5.50-$11.00
3617	Professional kiosk	$13.00-$26.00
3628	High-class professional	$32.00-$80.00

since the introduction of the Minitel. (After all, the PTT intended this development.) The increase of calls during school holidays certifies that games are especially appreciated. The young generation serves as Minitel's best public. All one can say is that France Telecom does nothing to prevent youngsters from abusing the system, but this explains why all present-generation Minitel terminals have a lock or a password.

After an initial boom phase, the *messageries* are levelling off. One reason is that the new terminals like Minitel 12 operate as sophisticated answering machines. You can now directly dial another Minitel terminal, read prepared messages, or leave yourself a message.

You can call your own Minitel terminal from another terminal to scan the messages or to change them. To give the same service to older Minitels, France Telecom operates a mailbox called Minicom, which is similar to IEEE Compmail+. The advantage of Minicom is that you pay electronic mail rates (independent of distance). For Minitel-to-Minitel conversation you pay normal (and possibly long-distance) telephone rates.

Minitel commercial services are increasing by 40 percent a year. The new projects include selling TV space and real estate and providing courses from a "teleuniversity." In addition, the Ministry of Industry has created the Euro 92 database to provide information about the progress of certain legislation and community projects.

The services market is very dynamic with a high turnover. Success requires a multichannel approach: advertising over TV or in the press, associating with other services, and offering alternate accesses to the service such as normal mail. Some people are beginning to worry about the potential disappearance of work for bank clerks and travel agents. Consulting bank accounts, making travel arrangements, and placing orders to warehouses over Minitel have become common. You can even pay bills directly with a smart-card reader connected to the Minitel terminal. With its elements of "teleshopping," electronic fund transfer, and gossip, Minitel has indeed become a marketplace in the original sense.

Meanwhile, Minitel has crossed some frontiers. SIP, Italy's state-owned operator, bought 300,000 Minitels for its own network called Videotel. The Spanish Bank of Santander distributed 40,000 terminals to its clients for remote banking. The US and Canada adopted the Teletel standard. Other countries follow a Prestel (English videotex) norm that resembles Teletel. They include the United Kingdom and commonwealth countries, Switzerland, Finland, and Denmark. Although Teletel and Prestel use the same transmission norm and similar display formats, you need a multistandard terminal to access either service. But you can access Minitel from almost all other countries (including the US) through a total of 150 gateways. Prestel countries offer both local and remote Minitel services such as the Suisstel in Switzerland.

Is Minitel a commercial success? France Telecom says not yet, arguing that its return on investment can't be measured for about a decade. But Minitel is already a social success, especially when compared with the sluggish start of similar systems in surrounding countries. Germany has less than 100,000 Bildschirmtext subscribers, and Switzerland only 24,000 videotex subscribers, in spite of large outlays for advertising. Except in France, the user must buy or rent the terminal separately. Who wants to spend US$550 for a simple directory? On the other hand, who wants to offer services to only 10,000 subscribers?

The French prefer to point out the technical superiority of Teletel, the innovative power of the service providers, and the quality of the Transpac PSDN (Public Switched Data Network) that made Minitel possible. The key idea, however, was to provide a free terminal and make money on the services. The principle is old. Back in 1910, Kodak sold its cameras at a bargain to make money on film, and Gillette gave away razors to sell blades. Today many mainframe representatives make a living from maintenance contracts. Nevertheless, the Minitel approach involved an element of chance. All the ingredients—the terminal, PSDN, services, and public demand—had to come together within the right time window.

The other European countries comfort themselves by regarding videotex as a transitory solution until the Integrated Services Digital Network arrives. ISDN plans to increase the data rate by a factor of 30 and offer additional flexibility such as document transfer and telefax replacement. Unfortunately, ISDN could stand for International Standardization Delays Networking. The progress is slower than anybody expected, and we are even uncertain that the final internetwork specifications of ISDN (the "blue book") will be available in 1990. In the meantime, the national societies develop their own versions of ISDN. We will have to live with bandwidth-wasting devices like telefax or Minitel that transmit digital data over analog channels to be redigitalized. ISDN will not make them obsolete. On the contrary, these devices offer today the services intended for tomorrow's ISDN.

Addresses

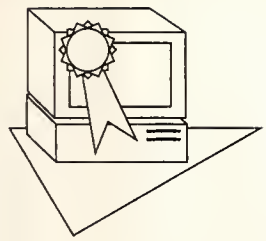
Minitel specifications
France Telecom
Direction Generale
6 Place d'Alleray
F-75740 Paris 15th District
France

Cnet Paris A
Documentation Technique
38/40 Avenue General Leclerc
F-92131 Issy Les Moulineaux
France

Minitel products
Telic/Alcatel
4 Rue de Chevilly
F-94267 Fresne Cedex
France

Transpac
Transpac, tour Maine-Montparnasse
33 Avenue due Maine, BP 145
F-75755 Paris Cedex 15
France

Department



New Products

Marlin H. Mickle
University of Pittsburgh

Send announcements of new microcomputer and microprocessor products, and products for review, to Managing Editor, IEEE Micro, *PO Box 3014, Los Alamitos, CA 90720-1264.*

Processors, Coprocessors, and Chips

DSP samples audio at 100-kHz

A floating-point DSP card for IBM PCs and compatibles processes digital and dual-channel audio analog I/O. Data can be converted to single-precision, IEEE Std. 754 format with one-cycle instructions.

Based on AT&T's DSP chip, the 50-MHz DSP-32C produces 25 Mflops and acknowledges existing AT&T development tools. In addition, the DSP card uses Motorola's DSP56ADC16 analog-to-digital converter to deliver a signal-to-noise ratio of 90 dB and a guaranteed THD+ noise of 0.005 percent.

Two 16-bit input channels permit simultaneous sampling of full audio-bandwidth stereo signals, with software-selectable sample rates up to 100 kHz per channel. Two 16-bit output channels feature concurrent output at sample rates up to 100 kHz. For applications needing to acquire or output large data blocks at full bandwidth, an on-board SCSI port can be connected to seven disk drives or other SCSI devices. **Ariel Corporation; $3,995.**

Reader Service No. 10

GaAs CPU operates at 50 MHz

Zero-wait-state performance up to 512 Kbytes can be found in a 50-MHz, gallium arsenide 68030 CPU. A SRAM-based board, the GMSV17-50M contains a 68882 floating-point coprocessor, two multiprotocol serial ports meeting RS-232/RS-422/485 specifications, and monitor/mailbox interrupt techniques that allow real-time multiprocessing. The double-Eurocard, single-slot board both originates and services interrupts on the VMEbus. Available operating systems include PDOS, PSOS, VRTX, and OS-9; additional software includes a debugger and on-board diagnostics. **General Micro Systems; from $6,895 (OEM prices).**

Reader Service No. 11

Accelerator provides 640 Mflops

The two-board CNP-3200, a 40-Mflops numerical processor when tightly coupled to a CPU, can be connected to other CNP-3200s to provide up to 640-Mflops performance. Designed for real-time applications such as simulation, signal and image processing, and oil exploration, the CNP-3200 runs with the CPU and one of the company's S-bus computers. One of the Fortran-programmable boards functions as the system NBI interface, and the other as the 8-Mbyte numerical processing unit. **Concurrent Computer Corporation; from $48,500.**

Reader Service No. 12

ISA system uses i486 processor

The i486-based Premium 486/25 ISA computer supports CAD/CAM, medical imaging, geophysical simulation, and aerodynamic stress-testing applications. In a multiuser environment, the desktop system can act as a server on a large network and run complex Unix/Xenix applications.

A standard 2-Mbyte, zero-wait-state configuration expands to 4 Mbytes on the processor board and 36 Mbytes on the system. **AST Research; from $8,495.**

Reader Service No. 13

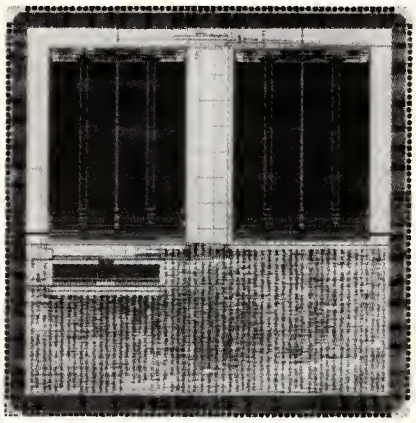
Parallel coprocessor features multiple transputers

The ProTran PC AT coprocessor features a T800 transputer-based parallel processing system; zero-wait-state performance; and optional, modified hypercube configurations with up to 16 interconnected transputers. With an expansion chassis, 32 transputers can be interconnected. Standard B004 software runs with the standard host MS-DOS or PC-DOS operating system so that all file system and operation interaction remains transparent to the user. **Yarc Systems Corporation; from $2,295.**

Reader Service No. 14

New Products

DSP chip identifies, tracks image objects



The L64290 Object Contour Tracer processes a 512 × 512-pixel image in 17 milliseconds (at a 15-MHz rate). Packaged in a 68-lead, ceramic PGA, the 1.0-μm HCMOS device performs in smart munitions, robotics, and machine vision applications.

The L64290 Object Contour Tracer uses a specialized processor and an internal memory that can store a 128 × 128-pixel image to identify and track objects in an image. The L64290 accepts input in the form of a binary image (1-bit-per-pixel data).

The 15- or 20-MHz chip outputs a sequence of values describing each contour in the input image, including the output's contour coordinates and a slope and curvature sequence. The device also computes a bounding box, the area, and the perimeter for each object in the image, to track a known target or object in successive image frames. **LSI Logic Corporation; from $296.**

15 MHz **Reader Service No. 15**
20 MHz **Reader Service No. 16**

Memory Updates

Configure SRAMs in two ways

A dual-architecture, 1-Mbyte CMOS static RAM lets designers select a configuration of either 1M × 1 or 256K × 4. The 35-ns M5M51001 serves as calculation-intensive mainframe and supercomputer main memory. It can also function as engineering-workstation and data-storage cache memory. **Mitsubishi Electronics America; from $195.**

Reader Service No. 17

Mask ROMs promise 200-ns access times

A 16-Mbit mask-programmable ROM permits high-density, one-chip storage with access times of up to 200 ns at 5V. Designed for increased software storage in word processor, hand-held computer, and video game products, the chip makes use of a flat-cell structure and a bank-select architecture. The flat cell measures 1.8 μm × 1.8 μm in a die size of 11.97 mm × 11.10 mm. **Sharp Electronics Microelectronics Division.**

Reader Service No. 18

Six densities offered in 2-Mbit ROMs

A family of six 110-ns ROMs, which includes a 2-Mbit device, feeds high-performance microprocessors without using wait states. The 2-Mbit part can store word processor, spreadsheet, and laser printer programs. The ROMs use a 5V power supply and consume 50 mA of operating current. **International Microelectronic Products; from $1.80 (10,000s).**

Reader Service No. 19

Reduce subsystem development cycles

R3000/3001 users can access a RISC subsystem family of modules to speed design cycles. One module, the IDT7RS501 MacStation development system lets embedded-control designers generate programs. They can also perform compilation and debugging in the company's Unix environment.

The MacStation requires a Macintosh II host with 8 Mbytes of main memory, an 80-Mbyte hard disk, and a 40-Mbyte tape drive for software installation.

A second embedded-control module is the IDT7RS102 CPU. This 3.2 × 3.9-inch CPU includes 16 Kbytes each of instruction and data caches. The CPU also features an optional 79R3010 floating-point accelerator.

An optional debugging tool for the CPU, the IDT7RS302 Target System development board, contains a 1-Mbyte main memory and a 256-Kbyte EPROM. **Integrated Device Technology; $6,900 (MacStation), from $1,045 (CPU), $3,300 (board).**

MacStation **Reader Service No. 22**
CPU **Reader Service No. 23**
Board **Reader Service No. 24**

Microcontrollers execute 5 MIPS

Two 8-bit, CMOS RISC microcontrollers, the PIC16C54 and -55, execute 5 MIPS in the design of data-entry products including computer peripherals and scanners. Each controller comes in EPROM and plastic, one-time-programmable versions for alteration of product features according to customer needs.

A security fuse in each device prevents stored programs or codes from being read from the finished product. **Microchip Technology; $2.40 (PIC16C54), $2.95 (PIC16C55)**

PIC16C54 **Reader Service No. 25**
PIC16C55 **Reader Service No. 26**

RISC News

R3000 controller processes 20 MIPS

The IDT 79R3001 RISController lets designers implement high-performance embedded-control systems based on the core MIPS RISC architecture. The 12.5- to 25-MHz 79R3001 increases the RISC synchronous memory space from 512 Kbytes to 16 Mbytes and processes up to 20 MIPS. **Integrated Device Technology; $120 (12.5 MHz), $299 (25 MHz).**

12.5 MHz **Reader Service No. 20**
25 MHz **Reader Service No. 21**

ASM devices include SRAMs

A second generation of application-specific memory, or ASM, devices includes 15- to 25-ns, synchronous fast static RAMs fabricated at 256K densities with 1.0-µm HCMOS technology. Additional ASM circuits include 20/25-ns, 64K × 1 × 4, synchronous parity RAMs, and 20- to 30-ns latched address RAMs. Plans call for introduction in the second and third quarters of 1990. **Motorola.**

SRAMs **Reader Service No. 27**
PRAMs **Reader Service No. 28**
Latched RAMs **Reader Service No. 29**

Parallel processor expands to 512 Kbytes of SRAM

A 50-MIPS development board for the IBM PC AT supports DSP, image processing, and graphics simulation and animation system design. The Forth-83 PP2000 board contains a 10-MHz RTX2000 RISC processor, a 32-Kbyte SRAM, a 386-Kbyte DRAM, and a 4-Kbyte dual-ported RAM. It expands to 512 Kbytes of SRAM. **C&C Technology Inc.; $5,575 (OEM discounts, 4 weeks ARO).**

Reader Service No. 30

Graphics system promises 128 MIPS, 192 Mflops

The Stardent 3000 uses a MIPS 32-MHz R3000/3010 scalar unit in a one- to four-processor configuration, each with a custom vector unit. Fully configured, the graphics system demonstrates a 105-Mflops rate for matrix multiplication in large linear equations.

The Stardent 3000 graphics system uses four 32-MHz R3000/R3010 processors to support a development organization with 128-MIPS and 192-Mflops performance.

Designed for scientific and engineering applications with interactive 3D graphics and imaging capability, the modular and expandable system can draw 50,000 to 150,000 color, 3D Gouraud-shaded quadrilaterals per second. Software includes Unix version 3 for multiprocessing; vectorizing and parallelizing Fortran, C, and Ada compilers; X Window System release 11.3; and OSF/Motif and PHIGS+ graphics tools. **Stardent Computer; from $69,000 (45 days ARO).**

Reader Service No. 31

CASE products

OS/2 tools generate designs, code

Two workstation capabilities for Texas Instruments Information Engineering Facility CASE products are in beta testing. One—an OS/2 version of the current MS-DOS workstation tools—aids planning, analysis, and design. The second is a code and database generation and transaction testing facility running under OS/2. The OS/2 version of the IEF allows users to design, generate, test, and maintain applications targeted for MVS, VM, IMS/DC, CICS, TSO, and PC batch applications. **Texas Instruments; beta version only.**

Tools **Reader Service No. 32**
Code generators **Reader Service No. 33**

OPS/MVS interface eases programming

Version 1.1.43 of the OPS/MVS interface lets users with no programming experience and no knowledge of programming languages create complex automation through its EasyRule facility. The menu-driven EasyRule guides a user through a series of panels that document and request information about the desired automation. From this set of answers, the facility produces fully documented REXX code to test and implement the automation.

Experienced users activate a shorthand method to explore new areas of the system and use an instrument to create program shells for later REXX augmentation. **MVS Software; from $9,500 (perpetual licenses, multiple-site discounts).**

Reader Service No. 34

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 **Medium** 181 **High** 182

Advertiser/Product Index

Coming in the June issue...

Hot Chips, Part 2

Articles from the IEEE Computer Society TCMM's Hot Chips Symposium continue with an 88000 family update, 8087 family stack problems, TI's floating-point unit, 68040, and the Endian Wars.

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northern California and Pacific Northwest: Roy McDonald Assoc. Inc., 5915 Hollis St., Emeryville, CA 94608; (415) 653-2122.
Jim Olsen, P.O. Box 696, Hillsboro, OR 97123; (503) 640-2011.

Southern California and Mountain States: Richard C. Faust Co., 24050 Madison St., Suite 100, Torrance, CA 90505; (213) 373-9604.

Midwest: The Kingwill Company, 4433 W. Touhy Ave., Suite 540, Lincolnwood, IL 60091; (708) 675-5755.

East Coast: Atlantic Representative Group, 349 Maple Place, Keyport, NJ 07735; (201) 739-1444.

New England: Arpin Associates, P.O. Box 6444, Holliston, MA 01746; (508) 429-8907.

Europe: Heinz J. Görgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G.); phone: (0 21 53) 8 99 88; telex 841(17)2153310=HJG tlx d.

Southwest/Southeast: Heidi Rex, Office, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

Moving?

PLEASE NOTIFY US 4 WEEKS IN ADVANCE

Name (Please Print) _____

New Address _____

City _____ State/Country _____ Zip _____

MAIL TO:
IEEE Computer Society
Circulation Dept.
10662 Los Vaqueros Circle
Los Alamitos, CA 90720

ATTACH LABEL HERE

- This notice of address change will apply to all IEEE publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

continued from p. 96

This example shows only one way that additional CPU power affects the transition to graphical user interfaces. Faster microprocessors made practical a shift from character-based, command-line-driven to bitmapped, graphical user interfaces. Future microprocessors will bring a new level of graphics capability to low-cost desktop computers.

Imagine a "room arranger" program that lets you move furniture around on a floor plan displayed on the screen. If this program were a simple, two-dimensional blueprint emulation, it could still be useful—but not very compelling. Suppose that three-dimensional images replace the outline drawings of furniture and that a three-dimensional view of the room replaces the flat, blueprint-like floor plan—from any desired perspective. Although such a program is much more useful, it raises the computing requirements by several orders of magnitude. As more and more computing power becomes available, the programmer can add details: fabric patterns for the couch and "daylighting" to show lighting variances during the day and the year.

Additional processing power would produce countless other advantages. Features such as automatic spell-checking while you type become more practical. Software programs are easier to write when you don't have to carefully optimize them for speed. Programs could perform more error checking and suggest intelligent alternatives instead of just responding with "unknown command." Emulation of foreign instruction sets and use of processor-independent intermediate languages would become more practical. Background processing could occur without noticeably slowing down the foreground task.

Business and home computer users desperately need computers that are easier to use. If the available computing power increases, programs could incorporate "intelligent agents" and other artifical intelligence like features that promise to make dealing with computers much more natural and less painful. Faster processors would also make speech recognition practical.

In the foreseeable future, software will become more useful with every increment of processor performance. New software will appear to take advantage of whatever amount of processing power becomes affordable—albeit with some time lag.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 186 **Medium** 187 **High** 188

Department

Micro View

Michael Slater
Microprocessor Report
(415) 494-2677
mslater@cup.portal.com

Who needs faster processors?

From the early Altair and Apple II computers to today's PCs and workstations, the power of desktop computers has grown by an unprecedented degree. A 25-megahertz, i486-based PC provides roughly 50 times the computing power of the original, 4.77-MHz, 8088-based IBM PC introduced less than 10 years ago. IBM's latest RISC-based systems provide more than double the performance of the fastest i486 system. While some experts don't expect performance to double every year, even the most conservative agree that performance will increase by at least a factor of 1.5 each year for the next few years.

With each generation of microprocessors, some observers always question the need for increased computing power. The introduction of Intel's i486, Motorola's 68040, and various RISC microprocessors unleashed a new wave of such commentary.

Almost everyone agrees that nearly infinite amounts of computing power can be put to good use in technical and scientific applications. Scientific visualization and simulations of all kinds constantly need more computing power. Software developers always want their programs to compile faster. The need for more computing power in business applications, however, is less clear.

Those who claim that typical computer users don't need any more processing power suffer from a lack of vision about what computers could be like. While much of today's software does not benefit from faster processors, the developer's assumptions about the speed of users' computers have inherently biased software design.

For a computer to seem responsive, it must respond to each keystroke in a fraction of a second. Studies have shown that user productivity continues to increase until the computer's response time drops to about 0.2 second. This phenomenon puts an upper limit on the number of instructions that can be executed in response to each keystroke. A 1-MIPS (million instructions per second) processor can execute only 200,000 instructions in 0.2 second. A 20-MIPS processor can execute 4 million instructions in the same amount of time. This increase makes possible an entirely new class of functions.

Even the most basic computer functions, such as word processing, become more useful and productive with a high-performance computer and software that is designed to take advantage of that performance. Microsoft Word was one of the first word processing packages for MS-DOS that used a bitmapped display (instead of a character-oriented one) to display italics, boldface, superscripts, etc. Processing dozens of pixels for each character—instead of a single byte representing its ASCII code—dramatically increases the burden on the processor. Word demonstrated that an 8088-based PC could indeed be too slow for word processing. Although Word was usable on early PCs, it was annoyingly sluggish. The IBM PC AT made Word into a reasonable performer, and an 80386-based PC makes most of Word's functions appear to be instantaneous.

As Word evolved, the company added features that raised the standard for the minimum acceptable processor speed. For example, the early version did not automatically repaginate documents. The amount of processing this task required after every keystroke would have made the program too sluggish on typical machines of that era. Now that the average machine is faster, this feature—among many others—has been added to the package.

continued on p. 95

0272-1732/90/0400-0096$01.00 © 1990 IEEE

IEEE COMPUTER SOCIETY

A member society of the Institute of Electrical and Electronics Engineers, Inc.

Use the Reader Service Card to obtain information on:

- Membership application—student #203, others #202
- Perodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Standards working groups list #195
- Compmail+ international electronic mail/database brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures—student/regular #193
- Student scholarship information #192
- Awards description/nomination forms #198
- Volunteer leaders/staff directory #196
- IEEE senior member application #204

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

Purpose

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

Membership

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others seriously interested in the computer field.

Publications and Activities

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and new products.

Periodicals. The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

European Office

Payments for Computer Society membership and publication orders are accepted by checks in Belgian, British, German, Swiss, or US currency. Checks in US funds must be drawn on a US bank. Payment may also be made by American Express, MasterCard, or Visa credit cards.

Asian Office

Payments for Computer Society membership and publication orders are accepted by checks in Japanese or US currency. Checks in US funds must be drawn on a US bank. Payment may also be made by electronic fund transfer to the Bank of Tokyo, Akasaka Branch, Toza acct. 0767956; the credit receiver is the IEEE Computer Society Headquarters Office. Payment may also be made by American Express, MasterCard, or Visa credit cards.

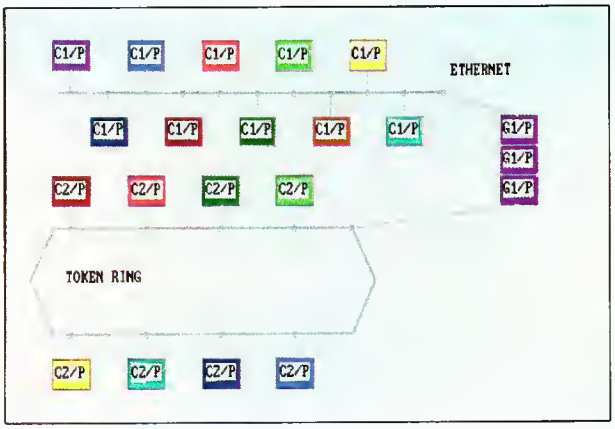
Ombudsman

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

New for network analysts and designers

Free trial and, if you act now, free training

SIMLAN II
Local Area Networks



SIMLAN II uses simulation to predict your LAN performance. You simply describe your LAN and workload.

Animated simulation follows immediately --no programming.

Easy-to-understand results

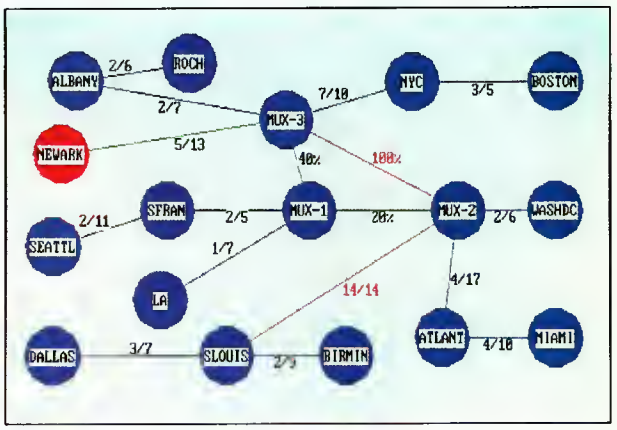
You get an animated picture of your LAN. System bottlenecks and changing levels of utilization are apparent.

Your reports show LAN statistics such as transfer times, delays, and queues. Client, server, and gateway statistics show queue lengths, waiting times, and messages sent.

Your LAN simulated

You can predict the performance of any LAN. Industry standard protocols such as Ethernet, Token Ring, and Token Bus are built-in. Variations can be modeled.

COMNET II.5
Wide Area Networks



COMNET II.5 uses simulation to predict your network performance. You simply describe your network, traffic load, and routing algorithms.

Animated simulation follows immediately --no programming.

Easy-to-understand results

You get an animated picture of your network. Routing choices and changing levels of network utilization are apparent.

Your reports show response times, blocking probabilities, call queueing and packet delays, network throughput, circuit group utilization, and circuit group queue statistics.

Your network simulated

You can predict the performance of any communication network--circuit, message, or packet switching. Virtual-circuit or datagram operation can be modeled.

Free 60 Day Trial Offer

The free trial contains everything you need to try SIMLAN II™ or COMNET II.5™ on your PC, Workstation, or Mainframe. Act now for free training--**no cost, no obligation.**

For immediate information

Call Chris LeBaron at (619) 457-9681, Fax (619) 457-1184. In Europe, call Nigel McNamara, in the UK, on (01) 332-0122, Fax (01) 332-0112. In Canada, call Francis Lobo at (613) 747-7467, Fax (613) 747-2224.

University faculty should call about our special offer for research and teaching.

Rush free trial & training information for:

☐SIMLAN II ☐COMNET II.5

Name

Organization

Address

Telephone Fax

Computer Operating System

Return to:
CACI Products Co.
3344 N. Torrey Pines Ct.
La Jolla, CA 92037
Call Chris LeBaron at (619) 457-9681.
Fax (619) 457-1184.

In Europe:
CACI Products Div.
Palm Ct., 4 Heron Sq.
Richmond-Upon-Thames
Surrey TW9 1EW, UK
Call Nigel McNamara on (01) 332-0122.
Fax (01) 332-0112.

In Canada:
CACI Products Co.
1545 Carling Ave.
Ottawa, Ontario
Canada K1Z 8P9
Call Francis Lobo at (613) 747-7467.
Fax (613) 747-2224.

IEEE MICRO